Question Answering

author names removed for a blind review

Important notes for editors/reviewers:

- We refrained from making any self-references for sake of a blind review. In the camera-ready version we would like to include them.
- Two topics are not discussed in the submitted version of the chapter. They are key elements of any QA system, but at the same time they are also a key element of many IR systems. Hence we assume that these topics are covered elsewhere in the book. If it is not the case we volunteer to extend our chapter with detailed discussion on these issues. The topics in question are:
  - NER – named entity recognition
  - GATE tool kit (General Architecture for Text Engineering)
- The section "Key terms & definitions" was moved to "Introduction" where it fits much better for this particular. If it is a problem from an editorial standpoint, we will adapt the chapter accordingly.
- We would love to provide information of the history of QA, mapping it from Turing, Green, Weizenbaum, Woods and others as surveyed by e.g. Simmons as well as the developments of Katz et al at the dawn of the Web but we eventually dropped this section because we understood from correspondence with the editors that it is preferred to focus on current and future issues.

ABSTRACT

Question Answering is an area of information retrieval with the added challenge of applying sophisticated techniques to identify the complex syntactic and semantic relationships present in text in order to provide a more sophisticated and satisfactory response to the user's information needs. As such it is widely accepted as the next step beyond standard information retrieval.

In this chapter state of the art of question answering is covered focusing on mapping of systems, techniques and approaches that are likely to be employed in the next generations of search engines. Special attention is paid to question answering using the World Wide Web as the data source and to question answering exploiting the possibilities of Semantic Web. Considerations about the current issues and prospects for promising future research are also provided.

INTRODUCTION

This chapter is dedicated to question answering (QA). We start with the motivation section where we explain the benefits of QA over the traditional keyword-based search. We also discuss the implications of the changing electronic market with particular attention to the boom of Internet-capable portable devices. Later we also present the commercial considerations of QA systems.

The main part of this chapter maps the state-of-the-art QA systems -- both research prototypes

and commercial products. During the mapping of the current QA landscape we cover all types of QA systems and describe systems of different scopes (open and closed domain systems) as well as of different levels of semantic processing (deep and shallow systems). We address various techniques used across all the systems with the emphasis on natural language processing and various statistical methods.

The objective of this chapter is to cover the technologies that are likely to be applied in the next generation search engines. For this reason we focus on two areas -- open-domain QA systems operating on unstructured text data (Web) and QA in the context of the semantic web.

Towards the end of this chapter we identify the problems and challenges that emerge as the current hot topics in the research community and/or have been reported as serious issues from the commercial sector.

OVERVIEW AND BACKGROUND

Question answering (QA) addresses the problem of finding answers to questions posed in natural language.

Traditionally the QA system is expected to provide one concise answer to the user's query. For the question "When did Thomas Jefferson die?" the ideal answer might be "July 4, 1826" with "Thomas Jefferson died on the Fourth of July, 1826" being another possibility. The exact way an answer is presented depends on the context and the application.

More formally, question answering is the task which when given a query in natural language, aims at finding one or more concise answers in the form of sentences or phrases. Due to its high requirements in terms of precision and conciseness, question answering is often seen as a sub-discipline of information retrieval (IR). Compared to IR, QA poses the added challenge of applying techniques developed in the field of natural language processing (NLP), such as the identification of the complex syntactic and semantic relationships present in the text.

QA systems even move a step further in natural language understanding with respect to standard IR systems (which have typical representatives in Web search engines) because they generally do not respond to a question but to a query in a form of a set of words where syntactic structure is ignored. Moreover, Web search engines do not return an answer, but rather a set of documents which are considered relevant to the query, i.e. which it is hoped will be useful to the user. Still, IR technology remains a fundamental building block of QA, in particular for those QA systems that use Web as their data collection (Quarteroni, 2007).

Motivation for Question Answering

Question answering (QA) is beneficial to users since it offers better user experience both in terms of relevance of provided information and of quality of user interface.

In case of delivering relevant information, QA systems benefit from advanced techniques for analysis of user queries which are capable of aggregation of partial results using mathematical operations, advanced comparisons, processing of temporal information and others. Moreover QA systems operating on Semantic Web can answer queries very precisely by transformation of questions to set of conditions used to generate logic query to knowledge base.

To highlight the benefits for user experience let us demonstrate the usability of QA systems vis-à-vis traditional keyword-based search engines with the following Web search example.

First, consider a scenario where the answer to a question is sought by a regular Internet user using a desktop computer. For many factual questions it is easy to find the answer very quickly using a conventional keyword-based search engine such as Google™. In this case the user is presented with the search engine return page (SERP) where the sought keywords are highlighted. By simply scrolling down the page and skimming the text close to keywords the user may often get the feeling of what is the right answer. Such a feeling is usually confirmed by random or deliberate clicks to a few top-ranked pages and assessment of the credibility of the presented information. This operation may take anywhere from as little as a few seconds to as long as a few minutes depending on whether the SERP provides enough trusted and/or easy to spot results.

Second, consider a user without access to a desktop computer. Instead let us assume she uses a phone or a personal digital assistant (PDA). The SERP as displayed on such a device shows much less information compared to a desktop device, which makes skimming the page cumbersome. Additionally, such devices are usually much slower then desktop computers, often harder to operate, and their lower bandwidth makes opening a few additional pages to confirm the credibility of an answer a rather painful experience. Such users would clearly appreciate a single sentence answer comprising just a few dozen bytes.

Users of cell phones with no Internet connection capability are left with SMS messaging only and have virtually no other option than to rely on single sentence answers. An extreme case would be users that prefer or must (e.g. due to impaired vision) make a regular phone call to a speech-based QA system.

With the current prevalence of the Internet connection capable mobile devices (Perez, 2008), the feature of being user-friendly (Budiu, & Hielsen, 2009) is becoming a more and more important aspect of the web search experience and it makes QA capability an essential competitive advantage of any future search engine.

Terms and Definitions

Question answering (QA) is the task which, given a query in natural language, aims at finding one or more concise answers in the form of sentences or phrases (Quarteroni, 2007).

QA is situated at the confluence of a large number of related areas (Maybury, 2004) including information retrieval (Gaizauskas, Hepple, and Greenwood 2004), natural language processing (Ravin, Prager, and Harabagiu 2001; de Rijke andWebber 2003), information extraction, and knowledge representation and reasoning (Harabagiu and Chaudhri 2002).

QA requires much more complex natural language processing techniques than other types of IR systems such as document retrieval. Natural language QA systems are thus often regarded as the next step beyond traditional Web search engines.

In general, QA systems are categorized according to two criteria -- the nature of data on which they operate and the level of semantic processing involved. These two criteria are orthogonal to each other (Figure 1).

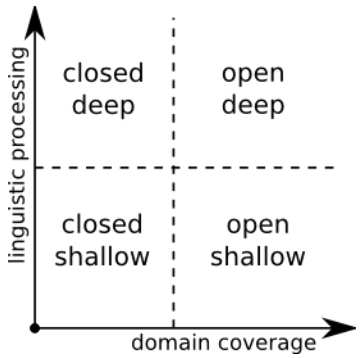

Figure 1: Classification of question answering systems.

In terms of the information source on which QA systems operate they are divided into closed-domain and open-domain.

Closed-domain QA deals with *questions within a specific domain* and can be seen as an easier task because natural language processing can exploit domain-specific knowledge frequently formalized in ontologies. Occasionally, closed-domain may refer to a *limited type of questions* that are accepted by a QA system, such as questions asking for descriptive rather than procedural information.

Open-domain QA, often called ODQA (Hori et al. 2003), appeared in the late 90's and soon became the standard in QA. In ODQA, the range of possible questions is not constrained, hence a much heavier challenge is placed on systems, as it is impossible to pre-compile all of the possible semantic structures appearing in a text (Quarteroni, 2007).

Recent research has moved away *from closed-domain* systems tailored to specific domains which (thanks to the possibility of using well-tuned but task specific question analysis techniques) do not offer enough scientific challenge, *towards open-domain* systems that can only rely on general knowledge which are very popular among researchers today.

In terms of methods and techniques the systems are divided by the level of understanding of the question into shallow and deep methods.

Shallow methods use local features for natural language processing. Local features are features that do not require sophisticated manipulating with context or building complex parsing structures. They include e.g. locating interesting snippets of text, detecting predefined patterns, matching with a set of templates or computing similarity to a defined set of questions. The shallow methods usually combine more local features together with the help of simple but robust statistics. The advantage of shallow methods is their robustness and reliability at the cost of failing to understand more complicated questions.

Deep methods on the contrary use more sophisticated linguistic processing to extract and construct the answer. They usually use context dependent analysis that may provide deeper insight into the user question but due to their high complexity they may fail more often than shallow methods.

The technical aspects of these standard approaches are discussed in detail in the next sections along with their various combinations, extensions and applications.

Forum for QA and Evaluation Framework

Before 1999 there was no dedicated forum to exchange ideas, positions and results in QA. Authors published their findings at random IR and NLP conferences and in journals, but it was only with the TREC-QA campaigns where researchers found their platform and QA has progressed in a major way.

The Text REtrieval Conference (TREC) organizes competitive tasks and comprehensive evaluation for natural language systems. From 1999 to 2007 TREC offered a QA track in which the task was to answer specific questions over a closed corpus. Apart from becoming a competition platform among QA systems it also has had a big impact on motivating their development. By defining new tasks for each year it also de facto formed the direction of research in QA.

Each year, TREC provided large-scale evaluation on increasingly difficult QA tasks, comparing systems from a growing community of research groups against a common metric, and raising the standards for the state of the art in QA (Bilotti, 2004).

The progress of the TREC conferences is well mapped in (Dang et al., 2007). Since its inception in TREC-8 in 1999, the QA track has steadily expanded both the type and difficulty of the questions asked. The first editions of the track focused on factoid questions. Whereas in TREC8, followed by TREC9 and 10, the QA system had to return the top 5 answers to the question, in TREC11 the response was limited to only one exact answer. In the following years (TREC12 and TREC13), the answer could be formulated as a string not exceeding 250 bytes. Moreover, systems competing in the TREC tasks must take into account other practical issues, as noted in (Harabagiu et al., 2003). These are *a large document collection* consisting of thousands of documents, *answer redundancy* because more sources can contain an answer for a certain question and *supplemental information needs*, when a document contains only a piece of the required answer.

The task in the TREC 2003 QA track contained list and definition questions in addition to factoid questions (Voorhees, 2003). A list question required different answer instances that satisfy the information need to be found in multiple documents, such as *List the names of whisky brands*. A definition question asks for explanatory information about a particular person or thing. Later the test sequence of questions was augmented with an explicit *"Other"* question, interpreted as *Tell me other interesting things about this as I don't know enough to ask directly*.

In TREC 2004, the target of questions could be a person, organization, or thing. Events were added as possible targets in TREC 2005, requiring that answers must be temporally correct. In TREC 2006, that requirement for sensitivity to temporal dependencies was made explicit in the distinction between locally and globally correct answers, so that answers for questions phrased in the present tense must not only be supported by the supporting document (locally correct), but must also be the most up-to-date answer in the document collection (globally correct).

The main task in the TREC 2007 QA track repeated the question series format, but with a significant change in the genre of the document collection. Instead of just news agency articles, the document collection contained also blog articles. Mining blogs for answers introduced significant new challenges in at least two aspects that are very important for real-world QA systems: (1) being able to handle language that is not well-formed, and 2) dealing with discourse structures that are more informal and less reliable than newswire. Based on its successful

application in TREC 2006 (Dang et al., 2007), the nugget pyramid evaluation method became the official evaluation method for the *"Other"* questions in TREC 2007.

The primary goal of the TREC 2007 main task (and what distinguished it from previous TREC QA tasks) was the introduction of blog text to encourage research in natural language processing (NLP) techniques that would handle ill-formed language and discourse structures. However, because most of the TREC 2007 questions requested factual information, they did not specifically test systems' abilities to process blog text and as a consequence answers still came predominantly from the newswire documents in the collection.

Because blogs naturally contain a large amount of opinions, it was decided by the organizers that the QA task for 2008 should focus on questions that ask about people's opinions and that there would be no factoid questions in later years (Dang et al., 2007).

Hence the TREC 2007 was the last QA track to date. In later years the focus shifted from QA towards (1) opinion seeking in the blogosphere with a larger collection of blogs and a much longer timespan allowing the temporal and chronological aspects of blogging to be investigated (Ounis et al., 2008) and (2) entity-related search on Web data where the entity is a person, product, or organization with a homepage where the homepage is considered to be the representative of that entity on the web (Balog et al., 2010).

The evaluation of QA system abilities has moved towards more complicated tasks, incorporating procedural questions, geographic reasoning (Santos, & Cabral, 2010), multilingual documents (Penas et al., 2010) or speech processing (Comas, & Turmo, 2009). Since the previous TREC tasks were oriented mostly towards newswire collections, the CLEF2009 forum decided to study whether the current state-of-the-art QA systems, mostly fine-tuned to the previous tasks, are able to adapt to a new domain and to move the QA field to more realistic scenarios. Furthermore, the CLEF 2009 firstly offered a multilingual fully-aligned question/answer corpus in eight languages (Bulgarian, English, French, German, Italian, Portuguese, Romanian and Spanish) to allow a comparison among systems working in different languages. The corpus contains five types of questions: factoid, definition, reason, purpose and procedure. Detailed descriptions of the tasks and evaluation of the results can be found in (Penas et al., 2010).

In TREC tasks, the standard measure for evaluating the performance of a QA system is the *mean reciprocal rank* (MRR). MRR is computed as follows: after putting the question into the QA system, a list of candidate answers is returned. The reciprocal rank for a given query *q* is *1/p(a)*, where *p(a)* is the position of the first correct answer within the returned list. If there is no such answer, the reciprocal rank is zero. The whole system is then measured as the mean of reciprocal ranks computed for each query.

Despite the overall positive effect of TREC to QA some criticized (De Boni 2004) the evaluation of the TREC-QA track pointing out the lack of a proper definition of the correct answer -- even in the case of factoid questions. For example, *What river is called Big Muddy?* for which the only accepted answer was Mississippi, although Mississippi River could also be considered as acceptable.

Another evaluation method may be used if a particular QA system requires a complete list of correct answers. In such cases, the concepts of *precision* (P), *recall* (R) and *F-measure* (F) are used as is common in IR. Let *C* be the number of correctly returned answers, *N* the total number of answers and *T* the number of all correct answers that should have been returned. Given

precision $P = \frac{C}{N}$ and recall $R = \frac{C}{T}$, the F-measure is computed as $F = \frac{2\,P\,R}{P+R}$. The general F_β measure can be expressed as $F_\beta = \frac{(1+\beta^2)\,P\,R}{\beta^2\,P\,R}$. In TREC2003, the Beta value was 5, indicating that recall was considered five times more important than precision (Voorhees, 2003).

STATE-OF-THE-ART QUESTION ANSWERING

In this section we provide overview of currently available technology used in QA. First we describe general architecture of current QA systems, latter we also discuss niche areas. Special attention is paid to systems operating on text data because these are the most common, the most advanced and also the most appealing for commercial sector. For these reasons they are the key candidate technology to be employed in the next generation of search engines which is the focal point of this book.

General Architecture of QA Systems

For a better understanding of the capabilities of a particular QA system, it is necessary to explore the types of questions it can handle. Generally, the two basic categories are *factoid* and *non-factoid questions* (sometimes simply called *why-questions*). Typical examples of factoid questions are "What currency is used in Ireland?", "When Thomas Jefferson died?" or "Who was the president of the United States in 1961?". These questions (simple, but not necessarily) can be answered by a short answer, e.g., date, name, location, etc. (so-called *named entity*). On the other hand, non-factoid questions may ask for reason, manner, method or definition and thus they require a more detailed explanation in the answer, e.g. a sentence or a paragraph.

Recently, attention to QA systems dealing with why-questions has risen. Although this type of question is not very frequent (e.g. 4.7% in the collection described by (Hovy, 2002), the research in this field is challenging since existing systems are not able to cope with this task using methods for factoid questions (Maybury, 2006). Moreover, advanced NLP techniques are essential for non-factoid question processing, both for understanding the question and answer extraction and formulation. Whereas factoid questions ask for a single piece of information and thus the answer is likely to be found in the documents explicitly, non-factoid question processing may involve semantic analysis and reasoning. The work of (Verberne, 2010) contains an extensive discussion about the issues of why-questions.

Moldovan et al. (2003) proposed a classification of questions and the appropriate QA systems into five classes according to their complexity.

- *Class 1* covers the already mentioned factual questions. The answer is presented directly in the retrieved text, it can be its morphological variant, or it can be extracted after simple keyword manipulation.
- *Class 2* QA system capabilities enable simple reasoning mechanism, such as semantic alternations or world knowledge axioms. These additional capabilities are necessary to answer questions that do not have a direct answer in the data collection.
- *Class 3* in the proposed classification covers QA systems which are able to fuse the answer from different documents.
- *Class 4* represents the interactive QA systems.
- *Class 5* covers QA systems capable of analogical reasoning or speculative question answering.

Although this formal QA system classification has not been widely accepted, it shows various levels of complexity and issues one has to face during QA system design and development. According to this classification, factoid questions were represented by only 67.5% in TREC8, TREC9 and TREC10, as described in (Moldovan et al., 2003). Over 27% were covered by questions of class 2 (requiring simple reasoning).

Regardless of the above mentioned query classification, most QA systems are built using a similar architecture. It provides a simple one-way dataflow and consists of three main modules: *question classification* (or question processing, query construction, etc.), *passage retrieval* (document retrieval) and *answer extraction* (answer formulation). See Fig. 2.

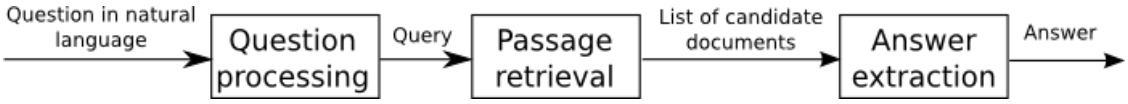

Fig. 2: Basic architecture of QA systems.

Some QA systems use finer module subdivision (i.e., 10 independent modules in (Moldovan et al., 2003) or 5 modules in (Moriceau & Tannier, 2010) but the basic structure remains. This modularity allows switching among particular implementations of certain parts of the system, as done for example with the passage retrieval module, where many systems use various off-the-shelf IR applications. The one-way data flow is also not mandated. Again, (Moldovan et al., 2003) tested architecture with feedbacks, resulting in a performance boost.

Question Processing

Generally, the purpose of this part of a QA system is to process the user questions expressed in natural language and transform it to any form which is suitable for the passage retrieval (PR) and answer extraction (AE) components. Although this step heavily depends on the target language and on particular semantic representation of the query, many QA systems use similar approach. Traditionally, two main outputs are produced by the question extraction component: the *expected answer type* and the *query*. The expected answer type describes what kind of information is being sought. The query usually contains question keywords used to retrieve documents with potential answers.

This step usually involves many NLP techniques for preprocessing, such as *tokenization*, *POS* (part-of-speech) tagging, *NER* (named entity recognition) or *syntactic parsing*. Some QA systems often attempt to extract the semantics from a sentence (Jurafsky & Martin, 2008) following the traditional human-computer dialogue best practices. In contrast, many QA systems do not require the full semantics (Verberne et al., 2010b), work only with a so-called *bag of words* and convert the question into a query by simply removing stop words and punctuation. A purely syntactic approach to question processing is used e.g. in (Moriceau & Tannier, 2010) which uses a robust parser for English and French.

Question Classification

Although *question classification* and *expected answer extraction* slightly differ in their meaning,

it usually describes the same task in most QA systems. Usually, its purpose is to determine the type of information which is required by the user. The expected answer type can be used not only as a constraint in limiting returned results in passage retrieval, but it is also important for answer extraction modules.

The question types may be a simple set (e.g. date, person, location, list, etc.) or they can be formed into a more complicated *taxonomy*. Only 5 types of question are considered by (Moriceau & Tannier, 2010), which are factoid (typically who, when, where questions), definition (e.g. "What is"), boolean (yes or no answer), complex question (why or how) and list. A more detailed classification, which involves also the question semantics rather than a simple question type, is introduced in (Quarteroni & Manandhar, 2008) where an 11 question type taxonomy was designed, consisting of e.g. human, geographical expression, organization, or temporal expression types for factoid questions and e.g. list of items, definition/description, or procedure for non-factoid questions. In (Buscaldi et al., 2010), a three-level taxonomy is proposed. On the first level, there are four main types: name, definition, date and quantity. On the second and the third level the types are more fine-grained, e.g. person, title, location, or acronym extend the name type on the second level, location is further divided to country or city on the third level.

Although named entity taxonomy is mostly flat (set of classes), hierarchical taxonomies allow more flexibility in matching answer type since an entity can be answered by its descendant entity (e.g. answer type *city* can be a valid for question type *place*, given such taxonomy). There is not a strong agreement whether richer taxonomy leads to less accuracy as claimed by e.g. (Kurata et al., 2004) or not as shown by (Laurent et al., 2005) with named entity topology consisting of 86 classes. In any case, answer type ambiguity due to overlapping types is an important issue. It can be handled either by allowing multiple type association or by applying the most specific type that covers all possible options (Verberne, 2010).

Assigning an input question to a question class can be viewed as an ordinary classification problem. Thus, many QA systems use standard machine learning algorithms for this task. In (Zhang & Lee, 2003) Support Vector Machines (SVM) were used with lexical, syntactical and semantic features. The SNoW model was used by (Li & Roth, 2006). Other features suitable for question classification were examined by (Quarteroni & Manandhar, 2008), including combination of bag-of-words, bigrams, trigrams, so-called bag-of-Named Entities and POS n-grams. A neural network-based classifier was used by (Comas & Turmo, 2009) where a rich set of lexical, syntactic and semantic features was fed to a multi-class perceptron classifier.

In contrast to the machine learning approach, answer types are also often identified using patterns or heuristics. For example, (Harabagiu et al., 2003) use a list of hard-coded rules to determine the required answer type from the output of syntactic parsing over the input question. Another pattern-based classifier, where the patterns are derived by analysis of the CLEF QA test set, is presented in (Buscaldi et al., 2010). These patterns are formed in a three level hierarchy and they are based on simple regular expression. Other similar approaches, e.g. (Voorhees, 2001b) search for specific pronouns and trigger words in order to extract the answer type.

Query Formulation

As mentioned previously, one of the main tasks of a question processing module is to produce a query which will be fed as input to a search engine. A query can be viewed as a set of keywords

or phrases which describe the user's information needs. In a traditional information retrieval system, keywords are more or less lexically related to the expected content of the returned documents (Manning et al., 2008). When dealing with a natural language question, there can be a wider stylistic gap between the input and the document containing an answer, especially when answering why-questions (as pointed out by (Soricut, & Brill, 2006)). Therefore, more attention must be paid to query formulation in order to improve the precision and recall of the passage retrieval system. The simplest approach, yet seldom used, e.g. in (Verberne et al., 2010; Correa et al., 2009), is to tokenize the input question and to remove stop-words and punctuation. Of course, this naive approach is far from being usable for questions which have an answer expressed using synonyms or for morphologically rich languages, where the answer can use the same words but their morphological categories differ. Another reason for using query reformulation is that the search engine yields better results if the query is formulated lexically similar to the answer and there is an exact match of phrases (Soricut, & Brill, 2006).

To deal with this issue, *query expansion*, *query reformulation* or *query transformation* techniques are used. As noted in the previous section, many QA systems use syntactic parsing and the semantics or the question keywords are extracted subsequently. To achieve a higher recall, the query can be enriched by morphological or semantic alternations of the keywords. This can include e.g. keyword lemma (using a morphological analyser) or keyword synonyms (using for example WordNet). This approach is used by (Molodovan et al., 2003).

The query reformulation technique is based on identifying various ways of expressing answer context given a natural language question (Kosseim, & Yousefi, 2008). It uses an assumption that the query can be expressed by a pattern, that could be formulated into a potential answer in the retrieved documents. One of the best results in the TREC10 task by (Soubbotin, & Soubbotin, 2001) was achieved by hand-written reformulation patterns. On the other hand, (Brill et al., 2001) used simple automatic permutations of the keywords for the same task, still getting reasonable results. This was mainly because of a large document collection, in which the answer can be found more than once and in various forms. Despite the fact that pattern-based query reformulation techniques can yield good performance when tailored to a specific language and domain, their main drawback is the high manual-development effort and a lack of portability.

Although many QA systems use some sort of pattern for query reformulation, more advanced NLP techniques were also involved. A full semantic representation of the question and the answer was tried by (Molla, 2009). Their system creates a graph-based logical form and the question answering is based on matching this graph-based representation. Nevertheless, producing such a representation is a very error-prone task. Thus, semantic features are often reduced to named entities, as e.g. in (Kosseim, & Yousefi, 2008).

A pure statistical approach to query formulation was introduced by (Soricut, & Brill, 2006). They observed that question reformulation does not have a positive impact when answering non-factoid questions, mostly due to the many possible ways of expressing the answer. For question transformation, they proposed a chunking technique based on co-occurrence statistics. Having a corpus of FAQ (frequently asked questions), they trained a statistical chunker on the answer set of the corpus in order to learn 2 and 3-word collocations. The difference between their chunker and a segmentation using a parser is that the phrases obtained from the chunker are not necessarily syntactic constituents.

Passage Retrieval

The basic assumption of QA systems operating on unstructured data is that the required answer is presented in a particular set of documents. The scale of such a set can vary from hundreds of documents (e.g. for intranets or closed-domain QA systems) up to the whole Web. For indexing and searching within such a quantity of documents, standard IR techniques are adopted by many QA systems. A comprehensive introduction to IR applications in QA as well as a detailed explanation of many state-of-the-art algorithms can be found in (Manning et al. 2008).

The simplest approaches to passage retrieval over the Web use existing commercial search engines, such as Google, Yahoo or MSNSearch. A significant advantage of such an approach is that these engines mostly use cutting edge technologies for indexing and retrieval and also they have the whole indexed Web available. Systems using passage retrieval based on commercial search engines are described e.g. in (Quarteroni & Manandhar, 2008; Soricut & Brill, 2006; Ifteen et al., 2010; Tannier & Moriceau, 2010).

On the other hand, question answering is a specialized task which differs from traditional IR in many aspects (among other things, in looking for keywords to obtain relevant passages instead of directly searching for an answer). Furthermore, major search engine companies make their profits through advertisements on their search pages and they do not offer any API of their services (i.e., the AJAX Search API from Google is not a standard web-service based API). To avoid these limitations, some QA systems adapt open-source search engines or commercial standalone search engine libraries. For example Apache Lucene (http://lucene.apache.org/) is used in (Moriceau & Tannier, 2010; Gloeckner & Pelzer, 2009) and as a baseline in (Buscaldi et al., 2010), or Lemur (http://www.lemurproject.org/) in (Verberne et al., 2010a).

When adapting a search engine, attention must be paid to proper selection of the indexing range. As mentioned first in (Harabagiu et al., 2003), three forms of indexing can be performed in QA systems. *Term or word-based indexing*, in its advanced form, includes multi-word term identifiers, document identifiers, and morphological, syntactic or semantic variants of the term, as used in (Ferret et al., 2001). *Conceptual indexing* involves a conceptual taxonomy that is built from the document collection and linked to the word-based index (Harabagiu et al., 2003). *Paragraph indexing* is based on the observation that the possible answer is likely to be located in the paragraph surrounding the keywords. It was implemented e.g. in (Harabagiu et al., 2000b). In later research by (Gomez et al., 2007) it was demonstrated that almost 90% of answers can be found in a passage consisting of three sentences.

For the searching procedure over the index, many passage retrieval components use the standard Boolean model (see e.g. (Manning et al., 2008)), Vector Space Model (see e.g. (Manning et al., 2008)), or Okapi BM25 (Beaulieu et al., 1997). From 11 participants in CLEF 2009, two competing systems used the Boolean model, whereas the rest mainly used VSM or Okapi (Penas et al., 2009). Unfortunately, the report does not show any apparent influence of the chosen model on the overall performance.

An extension of a traditional passage retrieval model towards QA specific information retrieval is presented in (Buscaldi et al., 2010). They present a passage retrieval system based on an n-gram model (clustered keyword positional distance model). In this system, an n-gram is a sequence of n adjacent terms extracted from a sentence or a question. The system is based on the premise that in a large document collection, question n-grams should appear more frequently near the possible answer. The answer coverage was compared with traditional IR methods (using

Lucene and IR-n (Llopis & Vicedo, 2002)), obtaining a 20% improvement on the CLEF 2005 test set.

The IR component may also be adapted to a specialized application. For example, to overcome possible drawbacks which can appear when dealing with automatic transcripts such as in CLEF 2009 QAST task (Comas, & Turmo, 2009), an IR engine relying on phonetic similarity can be employed as in (Comas & Turmo, 2009). It uses pattern matching algorithms to search for small sequences of phonemes (the keyword) in a larger sequence (the documents) using a measure of sound similarity.

Ranking

The list of documents or passages returned by an IR module is often sorted by IR score which is computed by the used retrieval model. However, the document with the highest score is not necessarily the document containing the desired answer. This is obviously due to the IR approach to passage retrieval itself. As mentioned before, strictly speaking, the traditional IR engines are not intended for question answering. Therefore, further *ranking* or *re-ranking* of the obtained passages is essential. Another reason for splitting the answer finding process into two parts, IR and ranking, is that IR operate on the whole document collections (which is mostly usually very large) and serves as a filter for selecting appropriate answer candidates that are subsequently subject to ranking. Since the document set retrieved by the IR module is then limited, ranking can involve more heavy-weight algorithms e.g. for NLP.

Whereas passage retrieval often uses existing off-the-shelf engines and algorithms, ranking modules are mostly application dependent. However, there are some common directions of research in candidate ranking, such as syntactic or semantic patterns, machine learning techniques, classifiers, etc.

A pattern based re-ranking approach is presented in (Kosseim, & Yousefi, 2008). In this work, the patterns are based on syntax and semantics and using the initial hand-crafted set of patterns, more patterns are generated automatically. After retrieving the top 200 candidates, the re-ranking is performed by measuring the similarity between semantic concept relations in the question and semantic concept relations in the candidate answers. The main drawbacks of this approach are that (1) the system requires a large collection in order to learn the patterns automatically and (2) it yields sufficient results only for factoid question because corresponding answers are expressed by a simpler pattern than is the case for non-factoid (why, how) questions. Furthermore, the system was developed and tested on the same type of corpus (TREC11) thus the performance result is likely to be much lower when adapted to different domain.

The hand-crafted syntactic rules are the core of another system introduced in (Moriceau, & Tannier, 2010). The set of 100 top documents is processed by a syntactic parser. Subsequently, about 40 rewriting rules are applied to obtain syntactic relations. Ranking is based on 9 complex heuristic rules which have been determined empirically. However, as conceded by the authors, this purely syntactic approach has some substantial disadvantages, such as the assumption that the documents are syntactically correct, or poor system speed (30 s per question) due to comprehensive parsing. This makes it unusable for practical deployment.

A ranking method based on machine learning is presented in (Verberne et al., 2010a). The authors aim at finding an optimal ranking function, having a set of features and various machine learning techniques (the problem is described as *learning-to-rank*). In QA systems, the list of

answers can be considered as a list of items described by a set of features and a class label, which determines the relevance of the item. The relevance is a binary function (the answer is either relevant or irrelevant) and the goal is to rank the correct answers higher than the incorrect answers. The tested methods also take into account the imbalance between the positive and negative instances in the training set, since there tend to be many more incorrect than correct answers (Usunier et al. 2004). Generally, the learning-to-rank can be viewed as supervised learning. The ranking function is trained using the given examples in the training stage to apply the ordered ranking in the testing stage. Authors of (Verberne et al., 2010a) name three different approaches to learning-to-rank. In the *pointwise approach*, the task of the classifier is to decide whether an answer is relevant or irrelevant to the given question, regardless of the relation among candidate answers to the same question. In the *pairwise approach*, the pairs of answers are mutually ranked within one cluster in order to optimize the proportion of the correct answers. More recently developed, the *listwise approach* uses an optimization of a cost function for the ordering of answers.

As for all machine learning applications, the choice of features is substantial. In (Verberne et al., 2009), and later used in (Verberne et al., 2010a), a set of 37 features is used by the ranking module. The first feature is the score returned by the IR engine. Further, syntactic features (such as subject, verb), WordNet expansion features, cue phrase features,document structure features and WordNet relatedness features are used. As machine learning algorithms, naive Bayess, support vector machines, support vector regression, logistic regression, ranking SVM, SVMmap and genetic algorithm were tested. Although the results are very promising and machine learning methods seem to handle well with imbalanced data, the authors conclude that their chosen features are only suboptimal for distinguishing correct from incorrect answers. A similar approach, based on learning-to-rank and features, can be found e.g. in (Higashinaka, & Isozaki, 2008). Their set of features is very large, consisting of 399 features in total. The features are: casual expression features (using automatically created patterns for casual expression from the EDR Japanese corpus), content similarity features (e.g. question candidate cosine similarity, question-document relevance, etc.) and casual relation feature. As in the previously mentioned system, the SVM ranking was used. Both above mentioned systems are focused on why-questions.

Answer Extraction

The task of the answer extraction (AE) module is to obtain the desired answer from the best-scored answer candidates and to present the proper formulation back to the user. The expression of the answer depends mostly on the question type. Since factoid questions ask for a simple fact (e.g. date, name, or other named entity), the answer containing only the named entity may be sufficient. In most cases, it depends on the QA system designer to decide whether the answer to a factoid question is expressed as one precise fact or as a whole sentence. Let us note that a limitation to one exact answer was a crucial requirement e.g. in TREC11 task. On the other hand, answers to non-factoid (why, how) question can be hard to express using a simple few-words answer and thus it is necessary to return a sentence or a whole paragraph that explains the complex answer to satisfy the user needs. Details about answer presentation are discussed later in this section.

Besides the ordered list of candidate answers, the input to the AE component includes the expected answer type and other constraints estimated during the question processing step. This

usually includes the answer target (e.g. a named entity type from a given taxonomy).

As in the question processing and ranking modules, the answer extraction approaches based on patterns or heuristics are used very often. Named entity recognition, POS tagging, parsing and other NLP techniques are also used very often as an answer preprocessing step. An example of a pattern matching approach is described e.g. in (Roussinov et al., 2008) or (Moldovan et al., 2003), where the patterns are hand-crafted. After filtering the retrieved passages regarding the expected answer type, the method assumes that the answer is presented in the document in a few exact forms and it can be extracted using templates and regular expressions. A combination of plain heuristics and similarity computing is presented in (Quarteroni, & Manandhar, 2008). For certain question types (e.g. time, money), class-specific rules are applied. For most factoid and non-factoid questions a bag-of-words similiarity is computed. This measure represents a number of matches between the keywords in the query and in the answer. Such simple score can also include features such as distance between keywords (Moldovan et al., 2003) or the occurrence of the candidate answer within an apposition (Pasca, & Harabagiu, 2001). A similar method, based on count of unique question keywords (and their variants or alternations, respectively) in the documents, served as a baseline in (Harabagiu et al., 2003). Furthermore, four different types of similarity were proposed by (Quarteroni, & Manandhar, 2008): (1) bigram similarity, which matches the common bigrams, (2) chunk similarity, where chunks produced by a shallow parser are measured, (3) head NP-VP-PP similarity, where the metric is based on a matching group consisting of a noun phrase (NP), verb phrase (VP) and prepositional phrase (PP), and (4) WordNet similarity, where the metrics exploit the WordNet lexical database and word-level distance from (Jiang, & Conrath, 1997). Similar five distances (word matching, WordNet matching, mismatch words, dispersion, and cluster words) were presented in (Ittycheriah et al., 2001) and used as an answer selection using maxiumum entropy. (Buscaldi et al., 2010) proposed another approach for choosing the best answer from top 'n' candidates, based on voting, where the candidates are compared by means of a partial string match.

Apparently, the pattern based approaches suffer from the necessity to create the patterns by a knowledge engineer manually. Thus, machine learning techniques were explored in order to eliminate the need of this effort and improve the ability to cover previously unseen questions. AE methods using simple machine learning techniques were discussed in (Harabagiu et al., 2000b) and later in (Harabagiu et al., 2003). The method is based on seven features and it learns a comparision function between candidate answers. These features share similar foundations with the previously described similarity distances. Instead of direct similarity computing, these features are used for training a classifier, using a labeled training corpus. As a classifier, perceptron was used by (Harabagiu et al., 2003).

A statistical model, based on the idea of a noisy channel, was presented in (Soricut, & Brill, 2006). This model consists of three modules, as shown in Fig. 3. The first module, *answer generation model*, proposes an answer *A* according to an answer generation probability distribution. The *answer/question translation model* further transform the answer *A* into the question *Q* according to conditional probability $p(A \mid Q)$. On the other hand, the task of *answer extraction module* is to find the answer which maximizes its probability given a question.

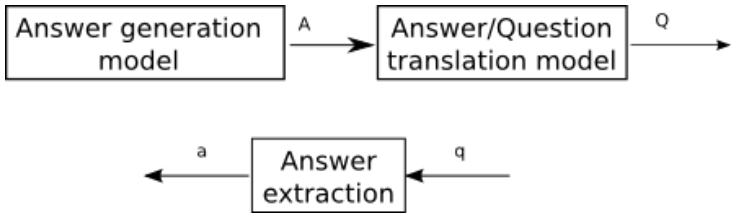

Fig. 3. A noisy-channel model for answer extraction, as proposed by (Soricut, & Brill, 2006)

Let 'T' be a task described as "find a 3-sentence answer for a given question". Then, we can formulate a *a-posteriory* probability of finding an answer *a*, given the question *q* and the task *T* as *p(a | q, T)*, which can be rewritten using Bayes' law to

$$p(a|q,T) = \frac{p(q|a,T)\, p(a|T)}{p(q|T)}$$

Since the denominator *p(q | T)* depends only on the task *T* and question *q*, it can be ignored. Then, the most likely answer that fulfills the maximal probability of being both well-formed and appropriate is

$$a = \text{argmax}_a\, p(a|T)\, p(q|a,T)$$

The first coefficient is question-independent, whereas the second one is question-dependent. This allows us to model the quality of a proposed answer *a* and the suitability of the answer *a* given the question *q* separately. To compute mapping between answers and questions, the correspondence between terms was used (called alignment). In (Soricut, & Brill, 2006), the model was trained on a question-answer corpus and the probabilities were computed using the expectation-minimization (EM) algorithm. The a-priori probability *p(a | T)* helps to downgrade inappropriate answers (e.g. too short or too long). A standard trigram language model was used to compute the probability distribution *p(. | T)*.

Answer Formulation

In non-dialogue based QA systems, the last step of the query life-cycle is the *answer formulation* and its presentation back to the user. This task involves many other aspects, such as answer credibility evaluation, proper answer formulation, and also so-called Web 2.0 services, such as user feedback, answer voting, etc. The answer presentation is probably one of the most crucial parts in commercial systems to satisfy user needs. It should assure the user that the system does understand the question properly and it should return appropriate formulation of the answer, given e.g. the question type. Although many academic QA systems achieve very good results in MMR, they do not pay much attention to the answer presentation step. This distinguishes them from many successful commercial systems.

As mentioned before, answers to factoid questions are often formulated only by a single word or other named entity. This limitation was also required in past TREC competitions. Furthermore, a concise answer is advantageous e.g. for devices with limited resources (bandwidth and screen size) such as smart phones. On the other hand, such a plain answer may not sufficient from the information credibility point of view and hence it should be supported by some additional

information to ensure the user that his question was well understood and that the data source can be trusted.

A simple method of providing a background for the answer is by presenting links to the top *n* documents which the answer was extracted from. The QA system can either provide the links to the documents or it can show snippets which contain e.g. keywords or the whole desired answer in a context. This approach is very similar to the one which is offered by current commercial search engines and it is also used e.g. in (Quarteroni, & Manandhar, 2008) or (Wenyin et al., 2009).

An extension of this approach can be found in QA systems, which use an advanced semantic processing of the question and the answer candidate documents. They are then able to formulate the answer as a complete sentence instead of a simple named entity. Apparently, the whole-sentence formulation has a positive impact on user satisfaction due to confirmation that the system understood the question properly (Kosseim et al., 2003).

Since the non-factoid question can be hard to answer by a named entity, the presented results mostly range from a single sentence to a whole paragraph.

Additional Supporting Information

Commercial QA systems, such as TrueKnowledge™ or WolframAlpha™, rely on a structured knowledge base instead of on textual data collection. Whereas QA systems operating on text data (typically Web) can only find their answers in textual form and present them as text, systems operating on proprietary datasets are able to enrich the answer with many additional explanations, e.g. graphics, formulas, or pictures. Furthermore, some systems can explain their inferring mechanism, e.g. which facts from its database were used or demonstrate their reasoning process. A set of possible related questions can be also provided.

User feedback is also a very important feature of successful commercial QA systems. If the user is satisfied with the given answer, many systems allows voting for such an answer (to increase its score). A QA system should be able to handle the searching failure, e.g. it does understand the question but it is not able to find an answer. It must be clear to the user, that this is not an issue related to the question and thus its reformulation does not yield any better results (Imiliensi, & Signorini, 2009).

Interactive QA: The Dialogue Approach

The lifecycle of a question/answer in traditional QA systems consists of a single step. After posing the question, the system returns either one precise answer or a list of documents, depending on the particular implementation. Due to the lack of any user feedback, the system must be able to understand the question properly and to serve the most precise answer. Hence, huge natural language question ambiguity is one of the issues that must be solved. To improve the results of QA systems, the whole process can be thus modelled as a dialogue, where in each following step the users' intentions are further clarified in order to obtain the answer to the right question. This variant of QA systems is called Interactive Question Answering (IQA).

Inspired by human information-seeking dialogue, (Quarteroni, & Manandhar, 2008) summarized the main issues that must be faced in IQA. These are e.g. *ellipsis*, when a dialogue participant omits a part of the sentence which is obvious from the previous context, *anaphoric reference*,

where an entity is not clearly formulated in the sentence but it is referenced from the dialogue context, *grounding and clarification*, when the systems puts the context into a clarification question to ensure the dialogue direction, and *turn taking*, which is apparently not very relevant in a textual dialogue. Given these possible issues, the requirements of the IQA dialogue manager must take into account e.g. context maintenance, utterance understanding given the previous context and mixed initiative, and so on. However, these issues are not in the core of this book. Instead, they relate to human-computer interaction and we invite the reader to consult e.g. (Ahrenberg et al., 1990) for detailed description of discourse representation and discourse management.

(Harabagiu et al., 2005) found that the quality of the IQA dialogue can be significantly improved when the system is able to predict a range of possible questions asked by the user. Their system, based on a framework called *predictive questioning*, uses a large question-answer database and it is used to propose a suggested question to the user. The suggested questions are selected according to the most salient aspects of the topic using 7 different similarity metrics. (Hickl et al., 2004) proposed question decomposition in a complex scenario environment in order to increase the performance of IQA. An ontology-based approach towards IQA is presented by (Magnini et al., 2009). In this work, the ontology is used for capturing aspects of interaction, such as constraints, question, context, etc.

Cross-language Question Answering Systems

Since 90's attempts to extend the knowledge base by incorporating documents from more than one language are regularly reported resulting in dedicated track within the TREC.

For translation two approaches are commonly used. They are based either on lexical resources (e.g., dictionaries, aligned word nets) or on machine translation (e.g., example–based translation). Role of machine translation was studied in (Larosa et al., 2005) concluding that extending the document collection by a new language improves the answering of only some domains of factual question. In contrary (and quite surprisingly) experiments documented in (Li, & Croft, 2001) indicate that techniques developed for question answering in English are also effective in Chinese thus demonstrating the techniques to be language neutral.

Considering machine translation the following three traditional approaches to cross language QA are used:

- translating the queries into the target language,
- translating the document collection into the source language or
- translating the queries and the documents into an intermediate representation (inter–lingua).

When dealing with huge amounts of data, as is the case for QA systems using Web as document collection, the only feasible approach today, is translating the question into the language of the document collection and the related issue of back-translating the answer into the language of the user.

Attention to cross-language aspects resulted in the development of a cross-language QA framework for both closed and open domains. The proposed framework (Sacaleanu, & Neumann, 2006) handles explicitly two aspects common to QA systems addressed -- cross-linguality (which is the original and prime objective of the framework) and credibility (which is

a general issue for all QA systems even those bound to a single language).

Recently an interesting experiment comparing QA in various languages was reported (Penas et al., 2009). This experiment differs from all the above in that here the prime goal is not to use documents in one language to answer questions posed in other language. Instead, systems competing in this experiment were working in different languages and were evaluated with the same questions over the same document collection manually translated into 8 different languages. It should be noted however, that the experiment was not aimed to provide simple concise answer to a question but rather to provide a single paragraph where the answer should be found by user himself. Evaluation of confidence in the answer was also considered by allowing the system to leave some questions unanswered which was scored as better then providing answer which is wrong. Organizers of the comparison identified many points for further improvement of their experiment promising more constructive conclusions in its future runs (Penas et al., 2009).

QUESTION ANSWERING IN THE SEMANTIC WEB ENVIRONMENT

In this section we provide an overview of the currently available technology used in QA systems exploiting the Semantic Web opportunities. First we describe a general architecture of such engines and later we examine particular implementation details. The Semantic Web is the second key element towards which the QA community is looking in the hope of new breakthroughs in understanding the information on the Web and thereby the ability to deliver the most relevant answers. Semantic Web technology thus forms the second element upon which the next generation search engines are likely to be built.

The *Semantic Web* vision is one in which the Web content is enriched with the semantic markup to allow machines to understand the meaning -- or *semantics* -- of information on the World Wide Web (Antoniou, & Harmelen, 2008). In this vision every piece of information is tagged (marked) and has a relation to an *ontology*. Automatic question answering engines can thus profit from such additional semantic information. Finding an answer to questions becomes a simpler matter if one can describe what is sought as a logic formula using the extra information stored in ontologies and semantic markup.

Semantic web tools use many technologies to process the Web content and various logic query languages to extract information -- e.g. SPARQL (Prud'hommeaux, & Seaborne, 2008) and SeRQL (Broeskstra, & Kampman, 2008). However, as the acronyms of the languages indicate, they are much more related to SQL than to a natural language. Hence the core task of question answering in the semantic web environment is to transform the question asked in the natural language to a logic language that can be used in the semantic web environment. In recent years many QA systems designs and prototypes have responded to this challenge. They usually share a common scheme which is presented in the following paragraphs.

QA systems exploiting the semantic web share some common features.

- They are closed domain systems -- they operate on one or more installed domains.
- They are easy to port -- the systems can be easily (at least in theory) installed on a different previously unknown domain with minimal effort needed.
- They use learning -- the systems learn from the ontology and also from user feedback.
- They use lightweight linguistic processing of the query -- see section "Lightweight

Syntactic Processing".
- Some of them also use a dialogue to clarify ambiguous relations.

The systems use two different knowledge sources for query analysis:

- knowledge of the natural language properties which are language specific and domain independent
- knowledge of the ontology which is domain specific and language independent.

The *knowledge of the natural language properties* means that systems know how to process the natural language. It contains the knowledge of morphology (part-of-speech tagging, lemmatization, stemming, ...), about the structure of a sentence (syntactic analysis, verb phrase chunking, ...) and the knowledge about the construction of a question meaning representation (general principles of semantic analysis).

The *knowledge of the ontology* allows the system to work with the semantics of the queries (semantic analysis). As we explained in the section "Elements of the Semantic Web" the ontology defines the meaning of concepts and the relations between concepts. We also noted that the ontology can be divided vertically according to levels and that the lowest level is too specific to be shared among domains. This makes these kinds of QA systems domain specific (and thus closed-domain). On the other hand, the ontology describes the semantics of the domain in such a way that QA systems can learn the rules for semantic analysis of questions automatically (see later in the section "System Architecture").

Elements of the Semantic Web

In this section a short description of the semantic web is provided. The purpose of this section is to ease the understanding of the following sections. This section focuses on the essence of the semantic web and abstains from description of every detail. For exhaustive technical details, refer to http://www.w3.org/standards/semanticweb/.

The purpose of the semantic web is to allow the content of the web to be understood and manipulated by computers. This will allow various tasks such as advanced information retrieval, knowledge aggregation and inference, information security control and so on to be automatically performed -- including question answering.

```
Notes for editors/reviewers:
      We assume to make references from the above paragraph to other related
      chapters of this book -- depending on its final structure.
```

In the semantic web the information (the knowledge) is stored in *triplets*. A triplet is a compound of a subject, a predicate and an object. A triplet represents a *relation* (predicate) between a subject and an object. For example, *[Beijing, isLocatedIn, China]*. The subjects and objects in triplets are called *concepts*. For more information about conceptual modeling, please see (Chen, 1999). All triplets are stored in a place called the *knowledge base*.

The *ontology* is a vehicle to describe the semantics of concepts and relations. The semantics is described by means of relations between concepts and relations themselves. A special type of a relation is the one called *isA* relation. This relation defines the taxonomy that is a relation between general and specific concepts or relations. For example, consider the following part of an ontology definition: *city isA municipality* and *municipality isA populated-place* etc. The

ontology may also specify that a city can be located in a state. In terms of our example ontology a relation called *locatedIn* may be defined between a *city* and *state* concepts. The principle we just have shown for concepts also applies to relations. For example, relation *locatedIn* is defined as *locatedIn isA geographicRelation.*

The ontology is sometimes divided into upper, middle and lower ontologies. The *upper ontology* (top-level ontology or foundation ontology) defines the most abstract concepts that can be shared by everyone. There are many upper ontologies e.g. WonderWeb foundational ontology (Masolo, et al., 2003), SUMO (Niles, 2001) etc. The *middle ontology* is more specific, more detailed and thus more difficult to agree on among stakeholders. It usually defines concepts that can be shared by a single system among all domains. The *lower ontology* is the most specific one. It is usually suitable for one particular domain only. The middle ontology has to be connected with an upper ontology and likewise the lower ontology has to be connected with a middle one.

There are two details worth mentioning. First, there are special types of concepts called *data values.* These are atomic values such as strings, integers, date/time values, etc. Data values can appear only at the third position (the object) in the triplet. The relation between a concept and a data value is called *data value relation.*

The second detail is that advanced versions of languages for ontology description allow *advanced properties of relations*, such as *transitive*, *symmetric*, *functional* and other properties to be described. These properties simplify the ontology definition and enrich its expressive power but they increase the demands on the ontology reasoning engine.

There are two essential technologies commonly used in the semantic web. The *Resource Description Framework* (*RDF*) is generally used to store triplets and for data interchange. The *Web Ontology Language* (*OWL*) is a language for authoring ontologies. These technologies built around W3C consortium are intended to provide formal description of concepts, terms, and relationships within a given knowledge domain.

Lightweight Syntactic Processing

Tools for a full scale syntactic analysis or other advanced means of linguistic processing are not usually used in semantic web applications and there are good reasons for this.

Arguably the most important reason is that the input queries are often not written as complete sentences or they are not grammatically correct. Because the input is only one short sentence or just a phrase there is not enough context to perform disambiguation required for more elaborate linguistic processing.

Effectiveness and sufficiency of lightweight syntactic processing is supported by (Katz, Lin, 2003) by showing that full syntactic parse trees capture relations and dependencies well, but they are difficult to manipulate with.

System Architecture

Despite some minor differences, all state-of-the-art systems share a similar abstract architecture. The six steps that are usually found in every QA system exploiting semantic web technologies are discussed below.

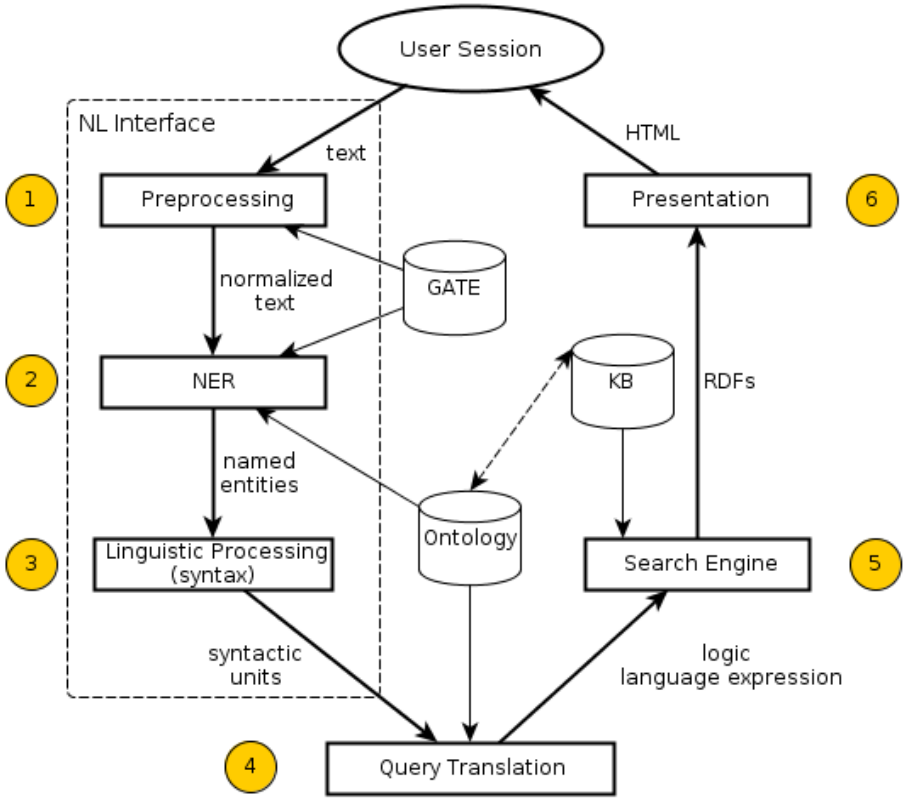

Figure 4: Abstract architecture of QA systems working within the semantic web framework

Figure 4 shows the general principle of operation of current QA systems.

Step 1

In the first step the system accepts a user input -– a question in the natural language. The input text is tokenized and preprocessed. During preprocessing stemming or part-of-speech tagging is usually performed.

Step 2

In the next step systems execute the NER task often with the help of *gazetteers*. A gazetteer is a large list of known words or phrases that need to be recognized in a text, these typically include various types of names, such as locations, organizations, or people and a variety of domain dependent terms -– see (Tablan et al., 2008) . The named entities are usually connected to the "bottom entities" in the ontology.

Step 3

The third step usually consists of some kind of lightweight syntactic processing. Common methods being used are the so called VP chunking (verb phrase chunking) or just a simple syntactic text chunking. Text chunking is a method of partial syntactic parsing. The method creates chunks that have no internal structure, in contrast to the phrases created during a full

scale syntactic parsing (Tjong et al., 2000). VP chunking is used e.g. in (Lopez et al., 2007).

Another method being used for lightweight linguistic processing is parsing with specially prepared context-free grammars taking advantage of the fact that the questions are being asked in a similar manner. Combined with the other means of linguistic processing mentioned in steps 1 and 2 and by using partial parsing, systems can reliably annotate syntactic information in the input query.

The techniques described above are the most often used, however we can find systems that are at both ends of the syntactic processing spectrum. For example, the system called FREyA described in (Damljanovic et al., 2010) uses full scale syntactic processing, while systems QuestIO (Tablan et al., 2008) and NLP-Reduce (Kaufmann et al., 2007) are at the other end of the spectrum since they do not use syntactic processing at all.

Step 4

The fourth step is the trickiest. It requires transforming the result of the syntactic processing to an expression in a logic language. The expression is then used for the actual search. So far, none of the current systems can cope with this problem without heuristic processing or without the help of the user.

The desired intermediate outputs of this step are the triplets that describe the question in the semantic web formalism. There two types of triplet. The first are those which have all three positions filled in. These triplets serve as constraints for the search. The second type are those which have one member of the triplet undefined. The undefined member is the point of the question that should be answered. The undefined member of the triplet is usually the subject (first member) or the object (third member) and very rarely the second member (predicate). When the triplets are created it is quite straightforward to build up the logic query for the search.

There are many approaches to creating the triplets that correspond to the question. Although the approaches may differ in many aspects, they share one key element -- the use of ontology *and a knowledge base*. As we explained earlier, the ontology describes concepts and relations between them. One triplet captures one relation between two concepts or between a concept and a primitive data type. Ontology defines exactly which concepts or data are related (or can be connected if one wishes to visualize it) i.e. which concepts and relations are legal values of a triplet. The concepts have names and the relation is also named. The systems use the assumption that the names of the concept and the relation are similar to the words used in the questions. For example, consider the following sequence of words placed in the question: "A city *is located* in a place" then the system assumes that there will be a relation named *isLocated* in the ontology that implements the link between a city (city is a concept in the ontology) and a place (also a concept). Obviously this assumption cannot be applied without an additional linguistic processing to handle an instance such as "A city *lies* somewhere" or "A city *is situated* somewhere", etc. There is a similar problem with the names of concepts. They can also be expressed in a different but synonymic expression or the name can be a compound of several words. This problem is however more frequent in the case of relation names than in the case of concept names. To address issues with different but semantically equal expressions, systems use dictionaries of synonymic words, string similarity techniques and learning. We address these methods in a separate section "Using Ontology as the Information Source".

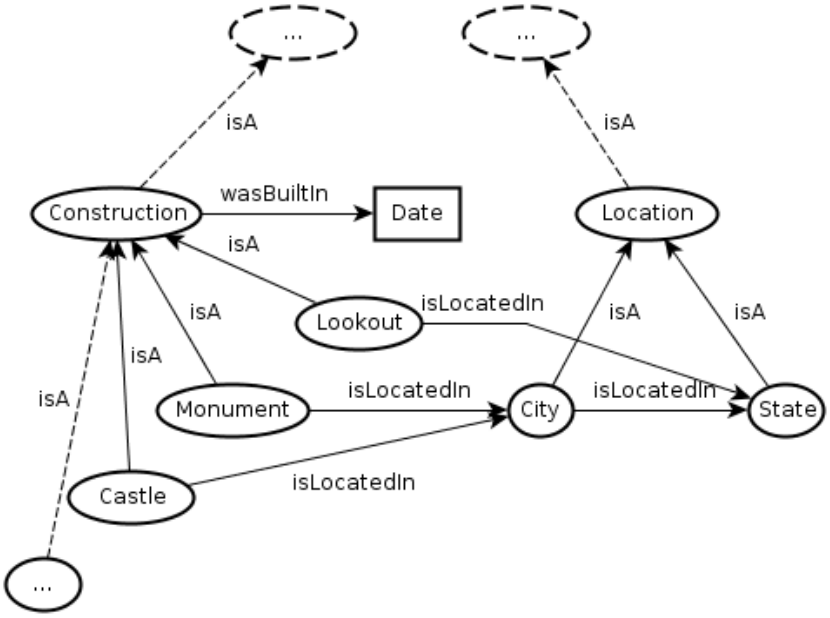

Figure 5: An ontology example.

The problem of expressions that are differently expressed but semantically equal is not the only issue to be handled. Many times it is also necessary to carry out advanced reasoning using the ontology. Let us demonstrate it using the simple example in Figure 5. Here the concepts of *city* and *state* are connected with the relation *isLocatedIn* however, the concepts of *monuments* and *state* are not connected. The relation though holds between those concepts as well (e.g. question "In what country lies the Statue of Liberty"). The reasoning process has to take into account that some relations have a transitive property. A similar kind of reasoning has to be done with inherited relations. If a relation holds between two superior (generic/general) concepts it also holds between two inferior (specific/concrete) concepts. For example, consider the relation *wasBuiltIn* that holds between concept *Construction* and the primitive data type *Date*. Clearly all superior concepts of the concept *Construction* can have this relation. Other types of advanced relations introduced in the section "Elements of the Semantic Web" have to be processed accordingly as well.

The last common problem is associated with *default relations*. The original query submitted by the user may contain generally known relations which are necessary to execute a successful search but that are not explicitly contained in the submitted text. These default relations have to be added during the process of creating triplets i.e. well ahead, before the search is executed.

During the process of creating triplets, an *ambiguity* may be encountered. In such a case some systems use a clarification dialogue. Usually such a dialogue presents different meanings in a list of choices or in a clarification question and the user is supposed to choose or answer. Some systems are able to learn user decisions, however the learning has to be treated with respect to the context and the user profile. The context is necessary because the same word can have different meanings in different contexts. Also, default relations may be ambiguous and valid only for a specific context.

The *user profile* is a special kind of context. Each user may use words with different meanings and also can expect different implicit knowledge of the system (internally represented by the

default relations). For example, consider the question "What is the largest river in Africa?" Here the word "largest" can mean *riverLength*, *riverWidth*, *riverBasin* or *riverDischarge*. After the user specifies that *riverLength* was meant then the search can be executed and the system learns this clarification with respect to the context and the user profile. The word *largest* itself can mean *riverBasin* for a different user. In the context of a different topic (for example, information about countries) the word can mean *stateArea* vis-à-vis for example *statePopulation*.

At this point the QA engine is ready to generate the logic query for the answer. All the triplets that have all positions defined serve as constrains. The triplet that has a position left free is the topic of the query.

Step 5

The resulting triplets from step four can be directly used as the logic query for the answer engine. The answer engine looks up the knowledge base and returns the sought information. Out of many formal query languages that are used to specify the search in a triplet database, two are the most often used:

- SPARQL: used in many RDF frameworks (Pellet, Jena, Redland, 3Store and others), W3C standard, used by (Kaufmann et al., 2007), (Damljanovic et al., 2010).
- SeRQL: has more SQL like syntax, used in Sesame framework, used by (Tablan et al., 2008).

The results of SPARQL and SeRQL queries can be *result sets* or *RDF graphs*. The *result sets* are tables with columns representing the language's SELECT statement (see (Prud'hommeaux, & Seaborne, 2008) or (Broeskstra, & Kampman, 2008)). RDF graphs are oriented graphs expressed in triplets. The triplets in the result graphs have the same format as the data definition in the knowledge base.

Step 6

The last step is a proper presentation of the found information. In the simplest case the information is presented as it is. Nevertheless, in more elaborate systems the information is presented within a grammatically correct sentence derived from the question. Some systems also provide an explanation of the search and inference processes to ensure the user that his or her question was correctly interpreted by the system. It also sometimes offers to browse the knowledge base for other relations.

This last step is in many aspects similar to the traditional QA systems described in the "Answer Formulation" section. It differs in how the answer can be explained and in the possibility to access the knowledge base through the ontology.

The explanation usually consists of showing the user how the logic query was constructed. Of course, it is not satisfactory to present the query in a SPARQL or SeRQL statement. It is more convenient for the user to show the triplet graph or to display the search constrains in a table (see (Damljanovic et al., 2010)). In more elaborate systems, users can alter the constrains to obtain different or more accurate answers (see (Cimiano et al., 2008)).

It is also possible to show the results in the context of the knowledge base. A user is allowed to browse other information related to the found answer. For example, consider the question "What is the largest river in Africa?" for which the system offers "Nile" as the correct answer. The user

can then browse other related information such as average discharge, states or cities through which the river Nile flows, and so on.

Ontology as the Information Source

The main domain specific knowledge source of the QA systems operating in the semantic web environment is the ontology. In this section we show how a system automatically extracts the information from ontology in order to minimize the system customization required when deployed to another domain.

Most of the time concepts and relations have meaningful names. The names are the main part of the information that is usable for query analysis. If a system finds in a question the named entity called *city* it looks for the concept *city* in the ontology. The same principle is applied to relations. When a system finds the verb phrase *is located in* it looks for the *isLocatedIn* relation. It is obvious that this basic approach would not be very robust if applied without any further "smart" processing when deploying the system to a new domain.

One vehicle used to improve the basic approach is the use of *language lexicons*. The lexicons organize words in synonymic groups and define relations (hyponymy, meronymy, etc.) between them -- (Jurafsky, & Martin, 2009). The use of the synonyms is obvious -- since a user can use a synonymic word in a query, the exact lexical match with the word stored in the ontology would fail without the use of synonyms. Hyponymy and meronymy relations work in a similar manner. A user can use a more abstract or more specific term than exists in the ontology.

Some systems make use of existing lexicons (such as the WordNet (Fellbaum, 1998)) others rely on lexicons created dynamically by means of learning, as the system is being used. The advantage of the former is that it is available beforehand, is usually well documented and has a proven record of successful deployments. The advantage of the latter is that it provides domain specific knowledge vis-à-vis the knowledge contained in off-the-shelf lexicons likely to be much more general and therefore not valid for a particular domain.

In the ontology the names of concepts and relations are usually written in the so called *pretty name notation*. This is the form of notation where words or phrases are demarcated by big letters or dashes instead of spaces -- e.g. relation name *isLocatedIn* or *is-located-in* is a phrase compounded from the words: *is*, *located*, and *in*. The algorithm that works with pretty names has to be able to provide approximate matches even when words are missing or left out.

The learning algorithm that extracts the knowledge from the ontology has to be able to get as much information as possible. A good practice when developing ontology is to provide extra description of concepts and relations. These are usually short texts that explain the details of concepts and relations that do not fit into their names. For example, the detailed description of the concept *monument* can be "Towers, statues, columns, archaeological sites and other structures of special interest". Such a description can be a rich source of useful words that extend the information contained solely in the name "monument".

Another way of providing a better match for words in questions with words in the ontology is to use string similarity metrics. The metrics can compare two strings and measure how alike they are. This is a good way of dealing with typos and slight differences in spelling. Some of the useful metrics are presented in (Cohen at el., 2003).

Naturally, all the above mentioned auxiliary techniques are not perfectly reliable and some sort

of scoring needs to be employed to make sure that a word that is exactly the same as the concept in the ontology is scored better than a word that is only matched by some of the techniques mentioned above.

Distinguished Features of Existing Systems

In this section we present several systems that use the architecture described above. Apart from the way the syntactic processing is implemented, these systems differ in some peculiar details that may be small but play an important role and make them worth mentioning.

Aqualog (Lopez et al., 2007) is arguably the best known QA system that uses the architecture described above. Its distinctive feature is that it uses triplets as an intermediate representation from the syntactic processing to the logic query transformation. When facing insufficient information, Aqualog exploits the user's help to disambiguate questions and learns user's hints. Although the authors claim Aqaulog to be easily portable there are some non-trivial procedures that have to be accomplished when deploying Aqualog to a new domain. First of all it has to be specified which terms "who", "where", "when" correspond to which concepts in the ontology. A second issue faced during installation is to choose or develop a suitable plug-in that maps names in the ontology (the *pretty names*) to the names used in plain text. A positive aspect of Aqualog is that it is a mature system that has been well tested with success of around 60% using the F-measure. Another important feature for the research community is the fact that it is developed as an open source project http://sourceforge.net/projects/aqualog and has a solid user base.

NLP-Reduce (Kaufmann et al., 2007) is a QA system that tries to reduce the NL interface functionality to a minimum. It does not rely on syntactic processing at all, with the sole exception of stemming, but NLP-Reduce uses WordNet to improve the process of generating ontology triplets. The advantage of the system is its very high level of portability at the cost of its question understanding being very shallow. The reported performance measured using the F-measure varies around 70% however the results are somewhat dubious.

The **Orakel** QA system (Cimiano, 2008) uses *logical description grammars* that combine syntactic rules for syntactic analysis and first order predicate calculus (FOPC) for semantic analysis. Named entities are used to generalize the FOPC rules, although it is not explicitly mentioned by authors. The advantage of the system is the compositional approach to semantic analysis. This way the system is able to cope with quantification, negation and conjunction in the question. The main drawback of the system is that it can cope neither with ungrammatical input nor with unknown words. Another weakness is the limited portability of the system because of being a complicated process -- the system requires a domain-specific lexicon to be manually created for each domain. To reduce the adaptation cost the graphical user interface for adaptation is offered. Interestingly, the adaptation is based on wrongly answered questions. The reported performance of the system in the F-measure metric is around 60%.

The **QuestIO** system (Tablan et al., 2008) is very similar to Aqualog although it lacks learning capability. Evaluation of its contribution is difficult due to the existence of a lot of hard-coded heuristic rules. Its value and distinctive feature is in putting more weight on the processing of incomplete or syntactically ill-formed questions by the extensive use of heuristics and scoring the intermediate results. The performance of the system was not given.

The **FreyA** (Damljanovic et al., 2010) is a follow-up research prototype based on the QuestIO system. It is one of the QA systems that exploits full syntactic processing with all the

consequences mentioned in the "Lightweight Syntactic Processing" section. The system uses advanced learning from the clarification dialogues called up in case of ambiguities. The learning involves not only aligning words from questions with words in ontology but also learning alternative scoring in the transformation process from syntactic units to logical query. The reported performance of FreyA is given in MRR. The system reached the value 0.72. However, the performance was measured on a small corpus of 250 questions.

ISSUES, CONTROVERSIES AND RECOMMENDATIONS

Reference Services and Knowledge Markets

In recent years we have seen a new type of on-line service rapidly gaining in popularity: the *knowledge markets* that are also known as *question-and-answer services* (Roush, 2006) or *question-answering communities* (Gazan, 2006). In essence these systems use Web 2.0 tools to implement the reference services of traditional libraries (in a traditional library there is a service on offer where the librarian answers a client's question or points him to resources).

Current computer-based solutions still involve human beings that answer questions. However, the solutions offer several levels of automation which we address in the rest of this section.

An example of the most straightforward implementation of such a human-assisted QA is the electronic reference and on-line library collection service "Internet Public Library" (IPL) (McCrea, 2004). IPL is a non-profit organization that relies on librarian volunteers and graduate students in library science programs to answer submitted questions. Technically this is not really interesting but it is seen as the baseline for the computerized human-assisted knowledge market QA systems described in the reminder of this section.

Web-based knowledge markets can be categorized into price-based and community-based services. The former includes Google Answers, Uclue and BitWine, while the latter includes Yahoo! Answers, and Knowledge-iN (KiN). A study (Chen et al., 2010) mapped the effects of price, tips, and reputation of the system on the quality of answers as well as on the answerer's motivation. This study led to such observations as offering higher prices for the answer leads to significantly longer, but not better, answers, while an answerer with a higher reputation offers significantly better answers.

A detailed description of technologies needed to implement a human-assisted QA system including *user modeling*, *reputation management*, and *answer clustering and fusion* is found in (Wenyin et al., 2009). The authors deal with the problem of handling accumulated answers in the system, which (if well archived and formalized in a global knowledge database) can be reused later. The idea of reuse is based on finding a match between the question asked by a user and an older question already answered and stored in the system. The main issue to be addressed is that the questions and answers in plain text format are not easily understood by machines and therefore it is difficult to find matches between questions.

The accumulated questions and answers are usually represented and stored in a pattern-based format, which can be converted to a formal knowledge representation for efficient machine understanding, processing, and reasoning. A *pattern-based user interface* for asking and answering questions was proposed in (Tianyong, et al., 2008) and (Wenyin et al., 2009) and named *semantic question pattern*.

In such systems the *model of user* who is answering questions typically includes

- A user's interest in the topic.
- A user's authority (capability of correctly answering the questions on a particular topic).
- A user's reputation which is a measure of how much he/she can be trusted when he/she promises something.

User reputation is derived from social network analysis where relational data is represented using a graph called a sociogram which is a directed and weighted graph (Wasserman, & Faust, 1994).

An interesting aspect of the system (Hao et al., 2008) is that even though the money offered for a certain question is zero, potential answerers also have motivations to provide their answers since there are chances that their answers can be automatically reused to answer new yet similar (or even exactly the same) questions in the future and they can still earn the money offered for such new similar questions.

Commercial Sector and Business Models

In recent years several commercial QA projects have been launched. The first commercially successful search engine with QA extension was arguably the Ask Jeeves (today known as Ask.com) founded in 1996. Ask Jeeves was sold in 2005 for $1.85 billion in stock showing clearly that question answering is a commercially attractive domain.

Since then several other QA systems have popped up. Notable projects were Yedda bought by AOL or PowerSet bought by Microsoft (Arrington, 2008). Other QA systems are being developed using venture capital worth tens of millions of dollars. A notable example is the promising prototype hakia.com (Weisenthal, 2007).

In general, little can be said about the technology used in commercial systems. For marketing reasons the description of systems and algorithms is usually boasted about, but superficial and in any case not detailed enough to verify any of the statements. Nevertheless, there are two systems that are worth commenting on.

The first service is *Wolfram Alpha* -- an answer engine developed by Wolfram Research, the company behind the popular Mathematica™ software. Wolfram Alpha is an online service that answers factual queries directly by computing the answer from structured data collected beforehand, rather than providing the answer from text documents or Web pages. The company is said to employ over 100 people to collect information on all the major branches of science. Thanks to the use of Mathematica™ this system is also well suited to questions involving algebra, symbolic and numerical computation, visualization, and statistical capabilities.

The second service is *True Knowledge* -- a QA engine using a database of discrete facts to comprehend posed questions by disambiguating from all possible meanings of the words in the question and to find the most likely meaning of the question being asked. The data are imported from "credible" external databases and from user submissions following a consistent format (Bradley, 2008). A distinctive feature of True Knowledge is that it offers the user not only the answer but also the facts and rules from which the answer is constructed.

Commercial QA systems use several rather distinct business models to make profit and/or attract users and/or investors. QA systems that require payment per answer are rare. Reputation-based

systems such as Yahoo Answers create revenue from advertising.

Some QA systems provide an API for a fee (Fiveash, 2009) and via this API it delivers answers to other applications. For example Wolfram Alpha delivers to Microsoft's Bing search engine. Moreover, some companies penetrate the mobile market such as iPhone and Android with paid applications to provide an optimized interface for mobile access.

An as yet unsolved issue is the question of using the intellectual property of third parties when delivering answers to QA system users. In the future the increasing quality of QA systems may lead to scenarios where users would merely use the QA engine and, being satisfied with the answers provided, they might never go the information source page. In this case the source page provider may lose revenues from advertising and in turn implement counter measures to prevent QA systems mining information from their web pages. A sustainable business model dealing with this problem has yet to be found.

FUTURE RESEARCH DIRECTIONS

As indicated earlier, QA is a commercially attractive domain and in all likelihood it will grow in importance in the near future. For Web-based QA there is still the issue of an unresolved business model relying on third-party information providers. It remains to be seen whether new services solutions would accelerate the QA domain even further.

Research interest in recent decades has moved from closed-domain systems to open-domain systems, namely to the Web. This shift has opened the field to new researchers resulting in never before seen investments in research both in terms of money and manpower.

The experience which the research community gathers while studying the open-domain systems may be applied back to closed-domain systems. These are traditionally perceived as a simpler problem due to limited data and the more controlled general knowledge which a system must possess to work effectively. However, with the advent of the semantic web and all its related technologies penetrating into QA, we may experience unprecedented applications of the technology developed for open-domain systems being used in their closed-domain counterparts, taking advantage of their constrained data.

As far as QA systems exploiting the semantic web technologies are concerned, the focus will be on harvesting the knowledge contained in the ontology to infer a sufficient amount of the information needed to perform the correct transformation of an input question to the logic query. By doing so, systems greatly increase their portability and reduce the difference between open-domain and closed-domain systems. With easy portability the closed-domain systems can be easily installed on another domain, thus covering more topics. In fact, some of the systems are looking into reducing the portability issues in order to be automatically able to cover all available domains in the semantic web framework. In this way the systems could achieve the properties of an open-domain system.

With the upcoming semantic web era (the so called Web 3.0), QA systems will be provided with extra possibilities for question answering. The pioneer systems working in the semantic web framework have shown the way, however many issues remain unsolved. Systems have yet to achieve the ability to work simultaneously with multiple resources (ontologies and knowledge bases). Knowledge bases with overlapping and possibly contradictory knowledge possess a

special challenge.

A hot topic is also the deep understanding of questions, especially covering negations, conjunctions, quantifications, comparison, temporal information and other semantically advanced features of questions. The ultimate goal is to combine deep understanding, processing of grammatically ill-formed or incomplete questions and domain independent processing.

Quite apart from the further evolution of the methods, techniques and algorithms already in place at various levels of maturity as described in the previous sections, there are still unsolved issues in QA that remain outside the scope of the current research but, in our opinion, deserve to be covered in the near future. We offer a short of overview of what we consider to be the "next big thing" in QA.

Assessment of the quality of answers to non-factoid questions is probably the most urgent issue given the recent gain in popularity of QA systems capable of answering these questions. In the near future we expect a lot of attention to be devoted to understanding the complex semantic relationships between parts of text corpora, presumably relying on tools such as FrameNet and PropBank (Boas 2009; Hwang 2010).

Ideally, a QA system should deliver correct answers knowing that they are correct, i.e., it can deliver a proof of the correctness of the answers. Since the current systems can only deliver answers with a certain trustworthiness (a credibility measure of an answer), determination of answer credibility may only be estimated from the plausibility of the answer source and context using various meta information. The first QA system credibility model (QAS-CM) was proposed by (Sacaleanu, & Neumann, 2006), but further research is needed before answer credibility is adopted by the industry, possibly complementing recent achievements in trust modeling (Netrvalova & Safarik 2008).

Research in personalized QA is still in a very early stage and a lot needs to be done at the modeling level. This means studying a user model and its attributes to represent the users of a QA system who can have very different expectations from the system due to their background knowledge (for example, a question about genetics asked by a medical student and by a child). It has already been identified as a key technology in (Maybury, 2002) but such a technique has not been applied to many QA systems since then, especially in open-domain QA (Komatani et al., 2003; Hickl & Harabagiu, 2006). A number of modeling issues were raised in (Quarteroni, 2007).

Notes for editors/reviewers:
We assume to make a reference here to another chapter of the book "Human-Centered Web Search" (depending on its content).

A related problem is the interactivity of a QA system and dialogue management strategy. This topic was also recently tackled in (Quarteroni, 2007), presenting a prototype implementation of a personalized and interactive QA system which may now serve as a baseline solution, but at the same time a number of open points were identified and called for further investigation. Commercial systems are arguably well ahead of academia in presenting the answers along with related information but the topic has not yet been rigorously studied in literature.

We expect that TREC will remain an important programmatic driver to facilitate the development of new technologies addressing specific needs as identified by the community.

CONCLUSION

This chapter covered the state-of-the-art of question answering. The focal point was in mapping of systems, techniques and approaches that are likely to be employed in the next generation of search engines. For this reason attention was paid to QA using World Wide Web as their data source and to QA exploiting the possibilities of Semantic Web. Later we provided considerations about the current issues and prospects for promising future research.

We have seen an enormous influx of resources -- both monetary and manpower -- into QA in recent years. Commitments announced by the search engine industry as well as steadily increasing activity in academia confirms that QA is an expanding research domain. We invite the reader to watch the topic carefully.

REFERENCES

- Moriceau, V., & Tannier, X. (2010). FIDJI: using syntax for validation answers in multiple documents. Information Retrieval, 13(5), 507-533.
- Soricut, R., & Brill, E. (2006). Automatic question answering using the web: Beyond the Factoid. Information Retrieval, 9(2), 191-206.
- Kosseim, L., & Yousefi, J. (2008). Improving the performance of question answering with semantically equivalent answer patterns. Data & Knowledge Engineering, 66, 63-67.
- Moldovan, D., Pasca, M., Harabagiu, S., & Surdenau, M. (2003). Performance Issues and Error Analysis in an Open-Domain Question Answering System. ACM Transcactions on Information Systems, 21(2), 133-154.
- Buscaldi, D., Rosso, P., Gómez-Soriano, J. M., & Sanchis, E. (2010). Answering questions with an n-gram based passage retrieval engine. Journal of Intelligent Information Systems, 34(2), 113-134.
- Wenyin, L., Hao, T., Chen, W., & Feng, M. (2009). A Web-Based Plaftorm for User-Interactive Question-Answering. World Wide Web, 12(2), 107-124.
- Verberne, S., Halteren, H., Theijssen, D., Raaijmakers, S., & Boves, L. (2010a, in press). Learning to rank for why-question answering. Information Retrieval, DOI: 10.1007/s10791-010-9136-6.
- Quarteroni, S., & Manandhar, S. (2008). Designing and Interactive Open-Domain Question Answering System. Natural Language Engineering, 15(1), 73-95.
- Verberne, S., Boves, L., Oostdijk, N., & Coppen, P. A. (2010b). What is no in the Bag of Words for Why-QA? Computational Linguistics, 36(2), 229-245.
- Higashinaka, R., & Isozaki, H. (2008, January). Corpus-based Question Answering for why-Questions. Paper presented at the Third International Joint Conference on Natural Language Processing, Hyderabad, India.
- Penas, A., Forner, P., Sutcliffe, R., Rodrigo, A., Forascu, C., Alegria, I., Giampiccolo, D., Moreau, N., & Osenova, P. (2010). Overview of ResPubliQA 2009: Question Answering Evaluation over European Legislation. In C. Peters, G. Di Nunzio, M. Kurimo, D. Mostefa, A. Penas, and G. Roda (Ed.), Lecture Notes in Computer Science: Vol. 6241.

Multilingual Information Access Evaluation I. Text Retrieval Experiments (pp. 174-196). Berlin, Heidelberg: Springer.

- Roussinov, D., Fan, W., & Robles-Flores, J. (2008). Beyond Keywords: Automated Question Answering on the Web. Communications of the ACM, 51(9), 60-65.
- Jurafsky, D., & Martin, J. H. (2008). Speech and Language Processing (2nd Edition). Upper Saddle River, New Jersey, USA: Prentice Hall.
- Beaulieu, M. M., Gatford, M., Huang, Y., Robertson, S. E., Walker, S., & Williams, P. (1997). Okapi at TREC-5. In D. K. Harman & E. M. Voorhees (Ed.) Information technology: The Fifth Text REtrieval Conference (TREC-5). Gaithersburg, MD: National Institute of Standards and Technology (pp 143-165).
- Harabagiu, S. M., Mariorano, S. J., & Pasca, M. A. (2003). Open-Domain Textual Question Answering Techniques. Natural Language Engineering, 9(3), 231-267.
- Varges, S., Weng, F., & Pon-Barry, H. (2008). Interactive question answering and constraint relaxation in spoken dialogue systems. Natural Language Engineering, 15(1), 9-30.
- De Boni, M., & Manandhar, S. (2005). Implementing Clarification Dialogues in Open Domain Question Answering. Natural Language Engineering, 11(4), 343-361.
- Imiliensi, A., & Signorini, A. (2009). If You Ask Nicely, I will Answer: Semantic Search and Today's Search Engines. In Proceedings of the 3rd IEEE International Conference on Semantic Computing (ICSC 2009), Berkeley, CA, USA: IEEE Computer Society (pp 184-191).
- Kurata, G., Okazaki, N., & Ishizuka, M. (2004). GDQA: Graph driven question answering system - NTCIR-4 QAC2 Experiments. In: Working Notes of the Fourth NTCIR Workshop Meeting, Tokyo, Japan (pp 338-344).
- Laurent, D., Séguéla, P., & Negre, S. (2005). Cross Lingual Question Answering using QRISTAL for CLEF 2005. In: Working Notes, CLEF Cross-Language Evaluation Forum (pp 21-23).
- Voorhees, M. (2001a). Overview of the TREC 2001 Question Answering Track. In Proceedings of the Tenth Text REtrieval Conference (TREC-10), NIST, Gaithersburg, Maryland, USA (pp 42-51).
- Soubbotin, M., & Soubbotin, S. (2001). Patterns of potential answer expression as clues to the right answers. In Proceedings of the Tenth Text REtrieval Conference (TREC-10), NIST, Gaithersburg, Maryland, USA (pp 175-182).
- Brill, E., Lin, J., Banko, M., Dumais, S., & Ng, A. (2001). Data-Intensive Question Answering, In Proceedings of the Tenth Text REtrieval Conference (TREC-10), NIST, Gaithersburg, Maryland, USA (pp 393-400).
- Ifteen, A., Trandabat, D., Moruz, A., Pistol, I., Husarciuc, M., & Cristea, D. (2010). Question Answering on English and Romanian Languages. In Peters, C., Nunzio, G., Kurimo, M., Mostefa, D., Penas, A., & Roda, G. (Eds.), Lecture Notes in Computer Science: Multilingual Information Access Evaluation I. Text Retrieval Experiments, vol.

6241, Springer Berlin / Heidelberg (pp. 229-236).

- Tannier, X., & Moriceau, V. (2010). Studying Syntactic Analysis in a QA System: FIDJI @ ResPubliQA'09. In Peters, C., Nunzio, G., Kurimo, M., Mostefa, D., Penas, A., & Roda, G. (Eds.), Lecture Notes in Computer Science: Multilingual Information Access Evaluation I. Text Retrieval Experiments, vol. 6241, Springer Berlin / Heidelberg (pp 237-244).
- Gloeckner, I., & Pelzer, B. (2009). The LogAnswer Project at CLEF. In: Working Notes for the CLEF 2009, Workshop, Corfu, Greece, September 30 - October 2, 2009.
- Ferret, O., Grau, B., Hurault-Plantet, M., Illouz, G., & Jacquemin, C. (2001). Terminological variants for document selection and question/answer matching. In Proceedings of the Workshop on Open-Domain Question Answering - Volume 12 (Toulouse, France). Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, (pp 1-8).
- Harabagiu, S., Pasca, M., & Maiorano, S. (2000b). Experiments with open-domain textual question answering. In Proceedings of the 18th Annual International Conference on Computational Linguistics (COLING-2000), (pp 292-298).
- Hovy, E. H., Hermjakob, U., & Ravichandran D. (2002). A Question/Answer Typology with Surface Text Patterns. In Proceedings of the Human Language Technology conference (HLT), San Diego, CA, USA, (pp 247-251).
- Maybury, M. (2006). New Directions In Question Answering. In Strzalkowski, T. and Harabagiu, S. (Eds), Text, Speech and Language Technology (32): Advances in Open Domain Question Answering, (pp 533-558), Springer, Netherlands.
- Verberne, S. (2010). In Search of the Why - Developing a system for answering why-questions. Doctoral dissertation, Radboud Universiteit Nijmegen, Netherlands.
- Zhang, D., & Lee, W. S. (2003). Question classification using support vector machines. In SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, Toronto, Canada, (pp. 26-32), ACM, New York, NY, USA.
- Li, X., & Roth, D. (2006). Learning question classifiers: the role of semantic information. Natural Language Engineering, 12(3), 229-249.
- Comas, P., & Turmo, J. (2009). Robust Question Answering for Speech Transcripts: UPC Experience in QAst 2008. In Peters, C., Deselaers, T., Ferro, N., Gonzalo, J., Jones, G., Kurimo, M., Mandl, T., Penas, A., & Petras, V. (Eds.), Lecture Notes in Computer Science: Evaluating Systems for Multilingual and Multimodal Information Access, Vol. 5706, Springer Berlin / Heidelberg, (pp. 492-499).
- Voorhees, M. (2001b). The TREC question answering track. Natural Language Engineering, 7, 361-378.
- Manning, C., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.
- Gomez, J. M., Buscaldi, D., Rosso, P., & Sanchis, E. (2007). JIRS Language-independent

Passage Retrieval system: A comparative study. In Proceedings of the 5th international conference on natural language processing (ICON-2007), 4-6 January. Hyderabad, India.

- Llopis, F., & Vicedo, J. (2002). IR-n: A Passage Retrieval System at CLEF-2001. In Peters, C., Braschler, M., Gonzalo, J., & Kluck, M. (Eds), Lecture Notes in Computer Science: Evaluation of Cross-Language Information Retrieval Systems (Vol. 2406), Springer Berlin / Heidelberg, (pp. 1211-1231).
- Larosa S., Penarrubia J., Rosso P., & Montes M. (2005). Cross-language Question Answering: The key role of translation. In: Proc. Avances en la Ciencia de la Computación, VI ENCuentro Int. de Computación, ENC-2005, Puebla, Mexico, September 26-30, (pp. 131-135).
- Li, X. & Croft, W. B. (2001). Evaluating Question Answering Techniques in Chinese. In Proceedings of Human Language Technology Conference (HLT-2001), San Diego, March 18-2, (pp. 201-206).
- Sacaleanu, B., & Neumann, G. (2006). Cross-cutting aspects of cross-language question answering systems. In MLQA '06: Proceedings of the Workshop on Multilingual Question Answering, Association for Computational Linguistics, Morristown, NJ, USA (pp. 15-22).
- Usunier, N., Amini, M., & Gallinari, P. (2004). Boosting Weak Ranking Functions to Enhance Passage Retrieval for Question Answering. In SIGIR 2004 workshop on Information Retrieval for Question Answering, (pp. 1-6).
- Verberne, S., Raaijmakers, S., Theijssen, D., & Boves, L. (2009). Learning to Rank Answers to Why-Questions. In Proceedings of the Dutch-Belgium Information Retrieval workshop (DIR 2009), (pp. 34-41).
- Pasca, M. A., & Harabagiu, S. M. (2001). High performance question/answering. In SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, New Orleans, Louisiana, USA. ACM, New York, NY, USA, (pp. 366-374).
- Jiang, J. J., & Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X), Taiwan, (pp. 19-33).
- Ittycheriah, A., Franz, M., Zhu, W., Ratnaparkhi, A., & Mammone, R. J. (2001). Question answering using maximum entropy components. In NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies, Pittsburgh, Pennsylvania. Association for Computational Linguistics, Morristown, NJ, USA, (pp. 1-7).
- Ahrenberg, L., Jönsson, A., & Dahlbäck N. (1990). Discourse Representation and Discourse Management for a Natural Language Dialogue System. In Proceedings of the Second Nordic Conference on Text Comprehension in Man and Machine, Taby, Stockholm.
- Harabagiu, S., Hickl, A., Lehmann, J., & Moldovan, D. (2005). Experiments with interactive question-answering, In ACL '05: Proceedings of the 43rd Annual Meeting on

Association for Computational Linguistics, Ann Arbor, Michigan, USA. Association for Computational Linguistics, Morristown, NJ, USA, (pp. 205-214).

- Hickl, A., Lehmann, J., Williams, J., & Harabagiu, S (2004). Experiments with Interactive Question Answering in Complex Scenarios. In Harabagiu, S., & Lacatusu, F. (Eds.): HLT-NAACL 2004: Workshop on Pragmatics of Question Answering, Boston, Massachusetts, USA. Association for Computational Linguistics, Morristown, NJ, USA, (pp. 60-69).
- Magnini, B., Speranza, M., & Kumar, V. (2009). Towards Interactive Question Answering: An Ontology-Based Approach. In ICSC '09: Proceedings of the 2009 IEEE International Conference on Semantic Computing. IEEE Computer Society, Washington, DC, USA, (pp. 612-617).
- Correa, S., Buscaldi, D., Rosso, P. (2009). NLEL-MAAT at CLEF-ResPubliQA. In: Working Notes for the CLEF 2009 Workshop, Corfu, Greece.
- Molla, D. (2009). From Minimal Logical Forms for Answer Extraction to Logical Graphs for Question Answering. Searching Answers: Festschrift in Honour of Michael Hess on the Occasion of His 60th Birthday, Münster: MV-Wissenschaft, (pp. 101-108).
- Voorhees, E. M. (2003). Overview of the TREC 2003 question answering track. In The Twelfth Text REtrieval Conference (TREC 2003), Gaithersburg, USA, (pp 54-68).
- Kosseim, L., Plamondon, L., & Guillemette, L. (2003). Answer Formulation for Question-Answering. In Xiang, Y. & Chaib-draa, B. (Eds.), Lecture Notes in Computer Science: Advances in Artificial Intelligence, Vol. 2671, Springer Berlin / Heidelberg, (pp. 983-994).
- Konopík, M. & Rohlík, O. (2010). Question Answering for Not Yet Semantic Web. In Sojka, P., Horák, A., Kopecek, I., & Pala, K. (Eds.), Lecture Notes in Computer Science: Text, Speech and Dialogue, Vol. 6231, Springer Berlin / Heidelberg, (pp. 125-132).
- Bilotti, M. W. (2004). Query Expansion Techniques for Question Answering. Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.
- Dang, H. T., Kelly, D. & Lin J. (2007). Overview of the TREC 2007 Question Answering Track. In Proceedings of the 16th Text REtreival Conference, Gaithersburg, Maryland.
- Ounis, I., Macdonald, C. & Soboroff, I. (2008). Overview of the TREC 2008 Blog track. In Proceedings of the 17th Text REtrieval Conference (TREC 2008).
- Balog, K., Vries, P., Serdyukov, P., & Thomas P. (2010). TREC Entity 2010 guidelines. Retrieved October 19, 2010, from http://ilps.science.uva.nl/trec-entity/guidelines/
- Santos, D., & Cabral, L. M. (2010). GikiCLEF: Expectations and lessons learned. In: Peters, C., et al. (Eds.) CLEF 2009 Workshop, Part I. LNCS, vol. 6241, Springer, Heidelberg (pp. 212–222).
- Masolo, C., Stefano B., Gangemi, A., Guarino, N., & Oltramari A. (2003). WonderWeb Deliverable D18. Technical report, Laboratory For Applied Ontology, ISTC-CNR, Trento, Italy.

- Niles I., & Pease A. (2001). Towards a standard upper ontology. In The 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Ogunquit, Maine, USA.
- Katz, B., & Lin, J. (2003). Selectively using relations to improve precision in question answering, in: Proceedings of the EACL-2003. Workshop on Natural Language Processing for Question Answering.
- Tablan, V., Damljanovic, D., & Bontcheva, K. (2008). A natural language query interface to structured information, Lecture Notes In Computer Science, in: Proceedings of the 5th European semantic web conference on The semantic web: research and applications, Spain.
- Lopez, V., Uren, V., Motta, E., & Pasin M. (2007). AquaLog: An ontology-driven question answering system for organizational semantic intranets, Web Semantics: Science, Services and Agents on the World Wide Web, Volume 5, Issue 2, (pp 72-105).
- Prud'hommeaux, E., & Seaborne A. (2008): SPARQL Query Language for RDF. Retrieved 15.10.2010 from http://www.w3.org/TR/rdf-sparql-query/.
- Antoniou, G., & Harmelen, F. (2008): A Semantic Web Primer, 2nd Edition, MIT Press, USA. March 2008. ISBN: 978-0-262-01242-3
- Broeskstra, J., Kampman, A. (2008): SeRQL: A Second Generation RDF Query Language, SWAD-Europe Workshop on Semantic Web Storage and Retrieval, 13--14 November 2003, Vrije Universiteit, Amsterdam, Netherlands.
- Chen, P., P., S. (1999): Conceptual modeling: current issues and future directions. Springer Berlin / Heidelberg.
- Tjong, E., F., Sang, K., & Buchholz, S. (2000): Introduction to the CoNLL-2000 Shared Task: Chunking. In: *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal, 2000.
- Kaufmann, E., Bernstein, A., & Fischer, L. (2007): NLP-Reduce: A "naive" but Domain-independent Natural Language Interface for Querying Ontologies", in proceedings of the 4th European Semantic Web Conference (ESWC 2007), Innsbruck, Austria.
- Damljanovic, D., Agatonovic M., & Cunningham H. (2010): Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction, In Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010), Springer Verlag, Heraklion, Greece.
- Cimiano, P., Haase, P., Heizmann, J., Mantel, M., & Studer, R. (2008): Towards portable natural language interfaces to knowledge bases - The case of the ORAKEL system, Data & Knowledge Engineering, v.65 n.2, p.325-354.
- Fellbaum, C. (1998): WordNet - An Electronic Lexical Database. MIT Press, USA. ISBN: 978-0-262-06197-1
- Cohen, W., W., Ravikumar, P., & Fienberg, S., E. (2003): A comparison of string distance metrics for name-matching tasks, pages 73--78.
- Jurafsky, D., & Martin, J., H. (2009): SPEECH and LANGUAGE PROCESSING: An

Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Second Edition, Pearson Prentice Hall. Pages 650-651.

- Roush, W. (2006). What Comes After Web 2.0?. MIT TechReview, retrieved October 18, 2010, from http://www.technologyreview.com/Infotech/17845/
- Gazan, R. (2006). Specialists and synthesists in a question answering community. Proceedings of the American Society for Information Science and Technology, 43(1), pp. 1-10.
- McCrea, R. (2004). Evaluation of two library-based and one expert reference service on the web. Library Review, 53(1), pp. 11-16.
- Chen, Y., Teck-Hua, H., & Yong-Mi, K. (2010). Knowledge Market Design: A Field Experiment at Google Answers. Journal of Public Economic Theory, 12(4), pp. 641-664.
- Wenyin, L., Tianyong. H., Chen, W., & Min, F. (2009). A Web-Based Platform for User-Interactive Question-Answering, World Wide Web, 12(2), pp. 107-124.
- Hao, T. Y., Hu, D. W., Liu, W. Y., & Zeng, Q. T. (2008). Semantic patterns for user-interactive question answering. Concurrency and Computation: Practice and Experience 20(7), pp. 783-799.
- Wasserman, S., & Faust, K. (1994). Social network analysis. Cambridge University Press, Cambridge.
- Arrington, M. (2008). Ok, Now It's Done. Microsoft To Acquire Powerset. Retreived October 18, from http://techcrunch.com/2008/07/01/ok-now-its-done-microsoft-to-acquire-powerset/
- Weisenthal, J. (2007). Hakia Raises $2 Million for Semantic Search. Retrieved October 18, from http://www.nytimes.com/paidcontent/PCORG_317848.html
- Bradley, P. (2008). True Knowledge - Questions Search Engine. Retrieved October 18, from http://philbradley.typepad.com/phil_bradleys_weblog/2008/09/true-knowledge---questions-search-engine.html
- Fiveash, K. (2009). Wolfram Alpha given keys to the Bingdom. Retrieved October 18, from http://www.theregister.co.uk/2009/11/12/bing_wolfram_alpha_deal/
- Tianyong, H.. Wanpeng, S., Dawei, H. & Wenyin, L. (2008). Automatic Generation of Semantic Patterns for User-Interactive Question Answering. Information Retrieval Technology, Lecture Notes in Computer Science, vol. 4993, Springer Berlin / Heidelberg, pp. 632-637.
- Perez, S. (2008). Mobile Web Use Growing Faster than Ever. Retrieved October 18, from http://www.readwriteweb.com/archives/mobile_web_use_growing_faster_than_ever.php
- Budiu, R., & Hielsen, J. (2009). Usability of Mobile Websites: 85 Design Guidelines for Improving Access to Web-Based Content and Services Through Mobile Devices. Nielsen Norman Group Report.
- Quarteroni, S. (2007). Advanced Techniques For Personalized, Interactive Question Answering. Doctoral dissertation, Department of Computer Science, The University of

York, York, UK.

- Naughton, M., Stokes, N., & Carthy, J. (2010). Sentence-level event classification in unstructured texts, Information Retrieval 13(2), pp. 132-156.
- Conesa, J., Storey, V. C., & Sugumaran, V. (2008). Improving web-query processing through semantic knowledge. Data & Knowledge Engineering 66(1), pp. 18-34.
- Blanke, T., & Lalmas, M. (2010). Specificity aboutness in XML retrieval, Information Retrieval, DOI: 10.1007/s10791-010-9144-6
- Paulheim, H., & Probst, F. (2010). Ontology-Enhanced User Interfaces: A Survey. International Journal on Semantic Web and Information Systems 6(2), pp 36-59.
- Horrocks, I., Patel-Schneider, P. F., & Harmelen, F. (2003). From SHIQ and RDF to OWL: the making of a Web Ontology Language. Web Semantics: Science, Services and Agents on the World Wide Web 1(1), pp. 7-26.
- Losada, D. (2010). Statistical query expansion for sentence retrieval and its effects on weak and strong queries, Information Retrieval, 13(5), pp. 485-506.
- Trotman, A., Geva, A., Kamps, J., Lalmas, M., & Murdock, V. (2010). Current research in focused retrieval and result aggregation, Information Retrieval, 13(5), pp. 407-411.
- Paris, C., Wan, S., & Thomas, P. (2010). Focused and aggregated search: a perspective from natural language generation, Information Retrieval, 13(5), pp. 434-459.
- Arvola, P., Kekäläinen, J., & Junkkari, M. (2010). Expected reading effort in focused retrieval evaluation, Information Retrieval, 13(5), pp. 460-484.
- Chengfei, L., Jianxin, L., Jeffrey, X. Y., & Rui, Z. (2010). Adaptive relaxation for querying heterogeneous XML data sources, Information Systems, 35(6), pp. 688-707.
- Trillo, R., Po, L., Ilarri, S., Bergamaschi, S., Mena, S. (2010). Using semantic techniques to access web data, Information Systems, DOI: 10.1016/j.is.2010.06.008
- Sánchez, D., Batet, M., Valls, A., & Gibert, K. (2009). Ontology-driven web-based semantic similarity, Journal of Intelligent Information Systems, DOI: 10.1007/s10844-009-0103-x
- Zhu, L., Ma, Q., Liu, C., Mao, G., & Yang, W. (2010). Semantic-distance based evaluation of ranking queries over relational databases, Journal of Intelligent Information Systems, DOI: 10.1007/s10844-009-0116-5
- Maybury, M. (2004). Question Answering: An Introduction. 3-18 Mark T. Maybury (Ed.): New Directions in Question Answering. AAAI Press 2004, ISBN 0-262-63304-3
- Gaizauskas, R., Greenwood & M., Hepple, M. (2004). Proceedings of the Workshop on Information retrieval for question answering at SIGIR 2004 workshop. SIGIR Forum 38(2): 41-44.
- Ravin, Y., Prager J., & Harabagiu, S., editors. (2001). Proceedings of the Workshop on Open-Domain Question Answering at ACL-01, Toulouse.
- de Rijke M. & Webber, B. editors (2003) Proceedings of the Workshop on Natural

Language Processing for Question Answering at EACL-03, Budapest*.

- Harabagiu, S. & Chaudhri, V. (2002). Proceedings of the AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases, Stanford.
- Hori, C., Hori, T. & Furui, S. (2003). Evaluation methods for automatic speech summarization. In Proc. Eurospeech 2003. (pp. 2825--2828).
- Boas, H. C. 2009. Multilingual FrameNets. In Computational Lexicography: Methods and Applications. Berlin: Mouton de Gruyter. pp. x+352
- Hwang, J. D., Bhatia, A., Bonial, C., Mansouri, A., Vaidya, A., Xue, N. & Palmer, M. 2010. PropBank Annotation of Multilingual Light Verb Constructions. In Proceedings of the LAW-ACL 2010. Uppsala.
- Netrvalova A. &, Safařík J. (2008). Selection of Partners for Co-operation Based on Interpersonal Trust. In: 2008 Conference on Human system interaction, Kraków.

ADDITIONAL READING SECTION

Although we have thoroughly supplemented the core sections of this chapter with references to available literature, some related issues may be of interest to the reader. Thus, we provide a list of recommended additional reading.

- Additional reading related to QA in the Semantic Web environment
    - A historical overview of the development of the Web Ontology Language (OWL), its incentives, fundamentals and philosophy is well described in (Horrocks et al., 2003).
    - Ontology-based QA using ontology databases, with application to two case studies in biomedicine, using ontologies and data from genetics and neuroscience can be found in (LePendu, & Dou, 2010).
    - A set of semantics techniques to group the results provided by a traditional search engine into categories defined by the different meanings of the input keywords is proposed in (Trillo et al., 2010). The authors claim that their proposal is different since their method considers the knowledge provided by ontologies available on the web in order to dynamically define the possible categories.
    - A very common problem in research areas such as natural language processing, knowledge acquisition, information retrieval or data mining is the estimation of the degree of semantic similarity/distance between concepts. (Sánchez et al., 2009) analyze this issue and propose modifications of classical similarity measures.
    - A complex survey of ontology-enhanced user interfaces can be found in (Paulheim, & Probst, 2010).
- Additional reading related to QA and IR
    - Sentence level classification is an important task for QA. Classification of unstructured text using SVM and language models can be found in (Naughton et al., 2010).
    - Incorporating semantic knowledge into query processing in traditional web search engines is described in (Conesa et al., 2008).

- The effect of distinct statistical expansion methods on sentence retrieval is thoroughly studied and evaluated in (Losada, 2010).
- Both focused retrieval and result aggregation provide the user with answers to their information needs, rather than just pointers to whole documents. An overview of the current research in this area is presented in (Trotman et al., 2010).
- Since aggregated search techniques represent an important part of the new generation of search applications, (Paris et al., 2010) focuses on exploring the parallels between aggregated search and natural language generation, leading to further advances in the way search technologies can better serve the user.
- An adaptation of ranking query to the relational database environment is presented in (Zhu et al., 2010) where the evaluation of ranking queries is based on semantic distance.

- Additional reading related to QA over structured data
  - A theoretical methodology to evaluate XML retrieval systems and their filters, along with the formal investigation of qualitative properties of retrieval models, is thoroughly described in (Blanke, & Lalmas, 2010).
  - A study presented in (Arvola et al., 2010) introduces a novel framework for evaluating passage and XML retrieval. This study seeks evaluation metrics for retrieval methods and proposes a framework, where the passages of the retrieved document are re-organized, so that the best matching passages are read first in sequential order.
  - XML query relaxation is necessary for searching XML data with a structured XML query, which can improve the precision of results compared with a keyword search. An adaptive relaxation approach which relaxes a query against different data sources differently based on their conformed schemas is proposed by (Chengfei et al., 2010).